REAL WORLD SECRET LEAKING

REAL WORLD SECRET LEAKING: THE DESIGN AND ANALYSIS OF A PROTOCOL CREATED FOR THE PURPOSE OF LEAKING DOCUMENTS UNDER SURVEILLANCE

BY

KARL KNOPF, B.A.Sc, B.Sc

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2019) McMaster University

(computing and software) Hamilton, Ontario, Canada

TITLE:	Real World Secret Leaking: The Design and Analysis of a Protocol Created for the Purpose of Leaking Documents under Surveillance
AUTHOR:	Karl Knopf B.A.Sc (Computer Science), McMaster University, Hamilton, Canada B.Sc (Statistical Science), University of Western Ontario, London, Canada
SUPERVISOR:	Dr. Reza Samavi and Dr. Douglas Stebila
NUMBER OF PAGES:	xiii, 115

Lay Abstract

Whistleblowing is an activity where an individual leaks some secrets about an organization to an unauthorized entity, often for moral or regulatory reasons. When doing so, the whistleblower is faced with the choice of acting publicly, and risking retribution or acting anonymously and risking not being believed. We have designed a protocol called the attested drop protocol, which protects the identity of the whistleblower, while allowing the unauthorized entity to have a means of verifying that the leak came from the organization. This protocol makes use of pre-existing identities associated with a communication medium, such as emails, to avoid using cryptographic primitives that are impractical.

Abstract

In scenarios where an individual wishes to leak confidential information to an unauthorized party, he may do so in a public or an anonymous way. When acting publicly a leaker exposes his identity, whereas acting anonymously a leaker can introduce doubts about the information's authenticity. Current solutions assume anonymity from everyone except a trusted third party or rely on the leaker possessing prior cryptographic keys, both of which are inadequate assumptions in real-world secret leaking scenarios.

In this research we present a system called the attested drop protocol which provides confidentiality for the leaker, while still allowing leaked documents to have their origins verified. The protocol relies on identities associated with common communication mediums, and seeks to avoid having the leaker carry out sophisticated cryptographic operations. We also present two constructions of the general protocol, where each is designed to protect against different forms of adversarial surveillance. We use ceremony analysis and other techniques from the provable security paradigm to formally describe and evaluate security goals for both constructions.

I would like to dedicate this thesis to my mother Eirin, my father George, and my brother Erik without whose support I could not have come this far.

Acknowledgements

I would first like to thank my supervisors, Dr. Reza Samavi and Dr. Douglas Stebila, for their contributions to this thesis and to my overall growth as an academic. Without them I would not have been able to complete this work, or have the confidence to pursue further education. I would also like to thank Dr. Håkon Jacobsen for his contributions to this work, as well as his support.

Next I would like to acknowledge my fellow graduate students at McMaster University. Whether we spent time in a research group, took classes together, or just talked over coffee, I appreciated every interaction.

Finally I would like to acknowledge the faculty members and staff at McMaster University for their contributions to my education. Thank you for providing a welcoming environment where I was able to learn and grow.

Contents

List of Figures

List of Tables

Notation and Abbreviations

Notation

$a \leftarrow B$	a is generated by algorithm B
$a \xleftarrow{\$} B$	a is randomly generated by algorithm B
$A \wedge B$	A and B
$A \vee B$	A or B
$a \notin B$	a is not a member of the set B
$a \in B$	a is a member of the set B
$a \notin B$	a is not a member of the set B
$\text{Exp}_b^{\mathbf{A}}$	Experiment $\mathbf{A}$, depending on parameters b
$\text{Adv}_b^{\mathbf{A}}(\mathcal{A})$	Advantage in $\text{Exp}_b^{\mathsf{A}}$ for algorithm $\mathcal{A}$
$\mathcal{A}^b(c)$	Algorithm $\mathcal{A}$ using oracle b and parameter c
$\mathsf{negl}(\lambda)$	A negligible function in security parameter λ. Alternatively, if for every positive polynomial $\text{poly}(\lambda)$: $\mathsf{negl}(\lambda) < \frac{1}{\text{poly}(\lambda)}$

DS	Digital Signature Scheme
$\mathsf{\Pi}$	Hash Function
H	Hash Algorithm
S	One-time Steganography Scheme
AC	Attestation Ceremony
G	The group of Identities
M	The set of generated messages
$l(i)$	The length of the object i in bits
$\perp$	Empty

Additional Notation

Provided in Subsection 2.3

Abbreviations

AD	Attested Drop Protocol
$\mathbf{ATT_i}$	Attestation Ceremony
DPG	Dropping Ceremony
$\mathbf{AP_1}$	Attestment Protocol One Sub-Ceremony
DLV	Delivery Sub-Ceremony

EXF	Exfiltration Sub-Ceremony
AP_2	Attestment Protocol Two Sub-Ceremony
DP	Drop Sub-Ceremony
VER	Verification Sub-Ceremony
PKI	Public Key Infrastructure
CA	Certificate Authority
KDF	Key Derivation Function
NIST	National Institute of Standards and Technology

Chapter 1

Introduction

In security research there has been a significant amount of work done on how to keep information secret from the public. There has been far less work done on the opposite problem, namely how to publicly reveal secret information. Secret leaking may seem like a trivial problem, as a user could just release the documents directly to a public forum, but this action could result in unwanted consequences for the user. Conversely, the leak could be done in an anonymous way, but then there would be no means of demonstrating that these documents are authentic.

1.1 Motivating Scenario

We motivate our research through the following scenario. An individual who is an employee of a government organization discovers evidence that a superior is embezzling money from public funds. With no faith that internal reporting systems will solve this problem, the employee seeks to 'blow the whistle' [NB74] on this abuse and release the evidence to the public with the hope that this will spur action from

elected officials. If he releases the documents anonymously, there could be doubts placed on the authenticity of their contents. So the individual seeks to release it to a reporter at a news agency hoping that they will provide some authority to the document. However the individual does not want to contact the reporter directly via email, as he does not want his identity to be revealed because he fears reprisal from his superior. The leaker seeks a system that allows him to both send the documents to a reporter anonymously but still be able to provide some proof that they are really from this government organization.

There are other similar scenarios where a solution to this hypothetical problem could resolve. A government researcher could want to leak documents to the press showing interference into public health research, but is afraid of being fired as a result [Aut19]. A physician could want to give federal investigators information about fraud committed by their peers, but is afraid of being excluded from an industry they spent their lives training to be a part of [Kol19]. A staff member for a major government figure could want to provide a statement to a news agency about how deranged the actions of their superior have become, but they fear serious reprisal if they identify themselves [Ano18].

1.2 Existing Solutions

The notion of a whistleblower, or someone who exposes illegal, incorrect and dishonest behaviour in an organization has existed for sometime, but it was not formally defined until Blackwell and Nader's book "Whistleblowing" [NB74]. The name refers to an individual who is blowing a figurative whistle, often a symbol used to warn others about potential dangers.

While this activity has not always been presented in positive light, public and corporate opinion on the idea has started to change. There are many corporate whistleblowing systems available for companies to use [DR, LT], that aim to let upper management be alerted to potential wrongdoings before they become public. In the public sphere, high profile cases, such as Edward Snowden's [Sch14], have led to increased awareness into the positive benefits of whistleblowing.

Traditionally, secret leaking has been done using live and dead drops. A live drop involves a direct communication, where the leaker hands the secret document to another party. A live drop is referred to in security research as a secret handshake protocol [AFNV19], as the identity of both parties will be protected from each other only if the handshake fails. A dead drop would have the leaker leave the information in a pre-defined location, allowing the other party to retrieve it at a later date [Rey06]. Both of these systems have been superseded by whistleblower-submission systems like Secure Drop [SPD] in the digital world. Here the leaker will create an anonymous communication with a drop server controlled by the other party, and deposit the documents there. None of these systems have a built in means of providing authentication to the documents sent by the leaker.

Another class of secret leaking systems rely on group and ring signatures. Group signatures were introduced by Chaum and van Heyst [Cv91], to allow a member of a group to be able to generate a signature on behalf of the whole group without immediately having to reveal which member was the signer. Since these constructions relied on a group manager and cooperation between group members to function, Rivest, Shamir and Tauman introduced the concept of ring signatures, in their paper "How to Leak a Secret" [RST01]. A ring signature scheme allows a signer to use

the public key of the other potential ring members to construct a signature with their own key pair, and can be done independently of any other member. Ring signatures have been worked on and extended [BKM09, SW07] for many years since, but all constructions still rely on the assumption that each group member is already associated with a pre-existing public key value.

A recent proposal to solve a similar problem is a blind CA [WAP+18]. Here, the system relies on an *anonymous proof of account ownership* to prove that the user belongs to an organization, without revealing their exact identity. This involves the user sending a message from their organization email to another outside email account, through a verifier, and using a *Secure Injection Protocol* to inject a challenge into the email. When the user recovers the challenge, he can then generate a certificate to be used in anonymous credential system [DFKP16]. While this system is able to generate an anonymous credential in its model, it does not consider an adversary who is recording communications from in and out of the organization. In such a scenario, the adversary would be able to look at past communications to find the individual who sent the message to the verifier thus immediately re-identifying the source. This type of attack is possible in a real-world secret leaking application, as an organization could record all communications if they are worried about this type of leakage.

1.3 Research Objective

The objective of this research is to develop a secret leaking system, which will be able to function in scenarios where a PKI is not available. In this system, the leaker will anonymously provide a group of potential leakers and their associated addresses to an attestor. The attestor will then send a challenge message to each member of

the group, where the actual leaker is only one to expect the message. The leaker will then use this challenge to complete a credential, which will allow him to provide some means of verifying the origin of any leaked documents to another skeptical party. The leaker may run this protocol with multiple attestors, but still use the same credential; this way the skeptic needs to only believe one of the attestors to accept the document's origin.

We formalize two of the goals of a secret leaking system using a reductionist security approach. The first goal is that given a list of potential leakers, the source is unable to be re-identified with a probability no better than random selection from the potential leaker list. We call this goal **Source Privacy**. The second goal is that any signatures created by a credential generated by the system cannot have been generated in another way. We call this property **Unforgeability**.

1.4 Methodology

We aim to prove the security of our constructions against specified attacks in a formal way. To do so, we use techniques that are part of an approach known as *provable security*, which is a well known methodology from cryptography.

1.4.1 Provable Security and Game-Hopping

Provable security. Provable security is the theoretical means by which modern cryptographic systems demonstrate that they are secure. Under this approach, a security proof consists of five parts: a definition of the model, a description of the protocol, a statement of the security goals and assumptions being made, a description

of the attack event, and a proof that the protocol meets the goals in the model [BWJM97], usually under some cryptographic assumptions.

There are two models used in this paper for definitions: the random oracle model and the standard model. The random oracle model provides to all parties in the protocol access to a truly random function. This function is provided as an oracle, meaning that no party can see how it operates but are able to call it when needed. The oracle is then used as a substitute for the hash functions in the protocol, allowing the proofs of security to make use of true randomness [BR93]. The standard model is where no additional assumptions are made and no additional oracles are provided.

Security goals are quantified by security experiments. Each experiment is defined as a game being played between the adversary and challenger algorithms. A challenger has three stages. First it will set up the necessary components for the experiment. It will then run the adversary, and interact with it as required. This interaction needs to be defined. The challenger will receive outputs from the adversary until it stops at its win condition, which was previously defined by the challenger.

There are many ways of realizing provable security, but our focus is on using an asymptotic-reductionist approach. Here we try to demonstrate that probabilistic polynomial-time attackers can only succeed a nominally small amount of the time, in relation to a previously specified security parameter. This is represented by arguing that the advantage an attacker has in the security experiment is negligible in respect to that parameter, often written as a function $\mathsf{negl}(\lambda)$. A function is considered to be negligible if it is asymptotically smaller than any inverse polynomial function [KL14, pg.48]. To prove that a function is negligible, we use techniques from computational complexity theory [CLRS09, pg.1067]. This approach was pioneered by Goldwasser

and Micali [GM84], with their work on developing semantic security models.

Provable security has become a common technique in modern cryptography, and appears in standard textbooks [KL14]. however it is not without controversy, as some believe that this approach provides a false sense of security for a protocol and allows its designers to ignore potential real-world attacks [KM07]. When used appropriately, and within the correct model [Bel98], provable security can provide a scientific means of gauging the abilities of a system based on cryptography [Dam07].

Game hopping Some of the proofs in this paper use game-hopping techniques to demonstrate security [Sho04]. A series of games are generated, labelled G_0 to G_n, where the goal is to show that each game is negligibly close to the previous game. This closeness can be shown through a set of operations known as transitions. In this work we use two types of transitions, those based on indistinguishability and those based on failure events. In the final game, we are able to directly reduce the security of the system to an underlying primitive and show that the adversary should not be able to win in this game. There exist other approaches to game based proofs [BR06], but they are not used in this work.

1.4.2 Ceremony Analysis

We examine the security of our solutions not just by using reductionist security, but also by a technique known as ceremony analysis. Ceremony analysis is an approach that aims to look at the context in which the protocol is used more than investigating the hard mathematical problems that support the protocol [RBGNB11]. It includes the human aspects of a protocol, which can be the source of security vulnerabilities that provable security approaches will overlook. By doing so, it can provide a more

holistic approach to investigating protocol's security.

The concept of ceremony analysis was first described by Walker, but was first used practically by Ellison to look at security vulnerabilities in HTTPS [Ell07]. He described the three properties of a security ceremony as: a ceremony is super-set of protocols; there is nothing out of band of the ceremony and; humans when part of the ceremony are explicitly included. By looking at these aspects, he describes Man-in-the-Middle attacks that a provable security approach would miss. Later works have used ceremony analysis to find vulnerabilities in the TLS protocol [RBGNB11] and in the e-voting protocol Helios [KZZ17].

To make use of security ceremony properties, we do the following in our analysis. We describe the protocol as a series of ceremonies and sub-ceremonies, while explicitly describing the relations between each entity involved. We include key generation and retrieval in our ceremonies to include all actions taken by actors in the ceremony in-band. In addition all human entities involved are explicitly included in the analysis of the protocol and we investigate the potential vulnerabilities they introduce. In summary we use ceremony analysis to describe potential issues within the protocol that a provable security approach would miss.

1.5 Contributions

In this thesis, we present the following major contributions. First we have developed a general protocol for secret leaking, which we call the attested drop protocol. This protocol relies on the leaker's ability to recover a challenge value sent to a group of organizational identities and provide support for the origin of the leaked documents. Second we present a means of evaluating secret leaking protocols, including our own,

against two goals: Source Privacy and Unforgeability. Finally we provide two constructions based on the attested drop protocol, where each construction is built for specific usability scenario. In the first scenario, the leaker is not under surveillance and they are able to extract large challenge values. In the second scenario, the leaker is under surveillance from the organization, so the challenge values sent must be easy to discreetly extract.

1.6 Outline

The rest of this thesis is organized as follows. In Chapter 2 we will discuss the necessary background and notations required to understand the following chapters. In Chapter 3 we will provide an outline of the general protocol. In Chapter 4 we describe the trust and threat models used to qualify the protocol. Next, we provide two instantiations of the general protocol and demonstrate that they meet the desired security and privacy goals. In Chapter 5 we provide a construction for unconstrained scenarios. In Chapter 6 we provide a construction for secret leaking scenarios in a monitored environment. In Chapter 7, we discuss various design options and future developments before concluding with a summary of the contents.

Chapter 2

Background

This chapter aims to provide the reader with the background material needed to understand what is presented in the subsequent chapters. Here we provide an introduction to various cryptographic topics and building blocks used in our solution. We then outline the notation we will use for our experiments.

2.1 Cryptographic Building Blocks

Public Key Infrastructure. Public key cryptography makes use of the fact that each party is able to generate a public-private key pair for use in protocols. But these keys do not have any identifying features, so one must trust that the public key they have is for the indented recipient. To solve this problem the public key infrastructure, or PKI, was proposed [Vac04]. In a PKI, a trusted third party called a certificate authority (CA) issues a certificate which cryptographically binds a public key to an identity. A PKI allows for its users to trust the claims of the public authority instead of having to blindly trust an anonymous party.

2.1.1 Digital Signatures

A digital signature scheme aims to provide authenticity to a message, by generating a verifiable signature for it. We are interested in when this is done in the public-key setting. Here the signer possesses a public-private key pair, and he uses the private key to generate the signature. Then anyone with the public key would be able to verify that the signature was associated with the corresponding private key.

Definition 1. *Adapted from [KL14, p.442]: A digital signature scheme consists of three probabilistic polynomial-time algorithms* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *such that:*

1. *The* ***key generation algorithm*** Gen *takes as input a security parameter* 1^λ *and outputs a public-private key pair* (pk, sk)*, each of length* λ*.*

2. *The* ***signing algorithm*** Sign *takes as input a private key* sk *and a message* m*. It outputs a signature* σ*.*

3. *The deterministic* ***verification algorithm*** Ver *takes as input a public key* pk*, a message* m *and a signature* σ*. It outputs a bit* b*, with* $b = 1$ *meaning* ***valid*** *and* $b = 0$ *meaning* ***invalid****.*

Signature Forgery. One security property that a digital signature scheme may want to achieve is unforgeability. A forgery is a message m, in conjunction with a signature σ that validates under the signature scheme's *Vrfy* algorithm with public key pk, but was not created by the holder of the corresponding sk. Figure 2.1 is a signature forgery experiment $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ to demonstrate that a digital signature scheme is *existentially unforgeable under an adaptive chosen message attack*, otherwise known as *secure*.

The signature – forgery experiment $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$

Adapted from [KL14, pg. 443]

1: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$

2: $(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(\cdot)$. Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its oracle.

3: **return** 1 iff $[\mathsf{Ver}_{pk}(m, \sigma) = 1] \wedge [m \notin \mathcal{Q}]$

Figure 2.1: The Signature Forgery Experiment

A signature scheme $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ in λ such that:

$$Pr[\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda} = 1] \leq \mathsf{negl}(\lambda)$$

2.1.2 Hash Functions

A hash function is a function that maps an arbitrary-length input to a fixed-length output. Given that hash functions can map from a larger input space to a smaller output space, there can be scenarios where at least two input values that map to the same output value. This is referred to as a collision, and a hash function is said to be collision-resistant, if it is infeasible to find a collision in polynomial time [Dam90].

Definition 2. *Adapted from [KL14, p.154]: A hash function (with output length l) is a pair of probabilistic polynomial-time algorithms $\Pi = (\mathsf{Gen}, \mathsf{H})$ such that:*

1. *The **key generation algorithm** Gen takes as input a security parameter 1^λ and outputs a key s, of length λ.*

2. *The* ***hashing algorithm*** H *takes as input a key* s *and a string* $x \in \{0,1\}^*$ *and outputs a string* $\mathsf{H}^s(x) \in \{0,1\}^{l(\lambda)}$

Hash functions are commonly used in cryptographic applications to build authentication schemes, such as digital signatures or message authentication codes. These functions are required to be collision-resistant, rather than simply being a recommendation for general hash functions [DY91]. In some situations other notions of security are used, namely preimage resistance and second-preimage resistance. Second-preimage resistance is considered a weaker notion of security, as collision-resistance implies it [KL14, pg. 156].

In practice hash functions are unkeyed, but for security definitions a keyed version is usually used. Any algorithm to find a collision could theoretically have hard coded values for an arbitrary collision, and just return these values in polynomial time. By forcing the key the hash function will use, we prevent this as an adversary would have to generate a prior colliding pair for each possible key [KL14, pg. 156].

Hash Collision Experiment. We define the hash-collision experiment $\mathrm{Exp}^{\mathsf{Coll}}_{\Pi,\lambda}$ for assessing the collision resistance of a hash function in Figure 2.2.

The hash – collision experiment $\mathrm{Exp}^{\mathsf{Coll}}_{\Pi,\lambda}$

Adapted from [KL14, pg. 155]

1: $s \leftarrow \mathsf{Gen}(1^\lambda)$

2: $(x, x') \leftarrow \mathcal{A}(s)$

3: **return** 1 if $x \neq x' \wedge \mathsf{H}^s(x) = \mathsf{H}^s(x')$

Figure 2.2: The Collision-Finding Experiment

A hash function $\Pi = (\mathsf{Gen}, \mathsf{H})$ is **collision resistant** if for all probabilistic polynomial time adversaries $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\lambda)$ in λ such that:

$$\Pr[\mathrm{Exp}^{\mathsf{Coll}}_{\Pi,\lambda} = 1] \leq \mathsf{negl}(\lambda)$$

2.1.3 Steganography

Core to the idea of protecting secret information is the notion of obfuscation, or the means to hide and confuse information from eavesdropping parties [BL18]. In contrast to cryptography, which seeks to obfuscate communications such that they are unreadable to outside parties, steganography seeks to prevent an adversary from realizing that there was any communications at all [HJ07].

Steganography is the practice of sending hidden messages through an insecure channel, such that an adversary is unable to detect them. This is typically done by creating a subliminal channel in the communication, where a hidden message is embedded in another public message [Sim83]. The set of messages that have been sent across the channel are referred to as a history. A steganographic scheme is considered to be useful if it is able to make any hidden message indistinguishable from the previous messages in the history.

Definition 3. *Adapted from [KRRS14]: A one-time steganographic scheme consists of three probabilistic polynomial-time algorithms* $\mathsf{S} = (\mathsf{SK}, \mathsf{SE}, \mathsf{SD})$ *such that:*

1. *The* ***key generation algorithm*** SK *takes as input a security parameter* 1^λ *and outputs a key* k *of length* λ*.*

2. *The* ***embedding procedure*** $\mathsf{SE}(k, m; \mathcal{O})$ *takes as input a key* k*, a message* m *of length* λ*, a history* h *and an oracle* $\mathcal{O}$ *that draws samples from the current history* $\mathcal{C}_{\langle}$*. It outputs a stegotext* $s \in \Sigma^*$*, where* Σ^* *is the possible message space*

3. *The* ***extraction algorithm*** $\mathsf{SD}(k, c)$ *takes as input a key* k *and some message* $c \in \Sigma^*$ *. It outputs either the message* m *or a token* ***fail****.*

Message Indistinguishability Experiment. A security goal all useful steganographic schemes aim to achieve is message-indistinguishability. This implies that any message sent by one of the parties that contains secret information, is indistinguishable to an observer from a message that does not. A one-time steganographic scheme $\mathsf{S} = (\mathsf{SK}, \mathsf{SE}, \mathsf{SD})$ is **secure** if for all probabilistic polynomial time adversaries SA_1, SA_2 there is a negligible function $\mathsf{negl}(\lambda)$ in λ such that:

$$\Pr[\mathrm{Exp}^{\mathsf{Mess\text{-}Indist}}_{\Pi,\lambda}(SA_1, SA_2) = 1] \leq \mathsf{negl}(\lambda)$$

2.2 Additional Building Blocks

Anonymous Communication. The establishment of an anonymous communication channel is an important part of many privacy oriented protocols and systems [LMP+12, KES+16]. The most commonly used software for creating these channels is Tor and its "Onion Routing" protocol [DMS04]. Onion routing involves creating layers of encryption for the message,and removing a layer at every hop in the message's route. By using a network of volunteer routers designed to carry out this protocol, Tor is able to protect communications against network surveillance.

There exist other protocols outside of Tor that are designed to create these channels. Vuvuzela [VDHLZZ15] hides the meta-data associated with instant messaging over the internet using techniques from differential privacy. It can hide who is communicating with who, but not that one is using the software. Loopix [PHE+17] is a message-based anonymous communication network that uses Poisson mix-nets to achieve third-party anonymity. This implies that an adversary can only observe that someone has used Loopix, but not who they are communicating with.

2.3 Notation for Security Analysis

To describe the protocols and ceremony we present in this thesis, we use a set of notation that builds upon existing work [KL14] but is designed to meet our needs. We use this notation to describe the various algorithms, oracles and protocols we design. Using it, we are also able to succinctly demonstrate which sections of the protocols an adversary is able to interact with and control.

Algorithms and oracles. Let B be an (possibly probabilistic) algorithm. The notation $y \leftarrow B(x)$ denotes running B with input x, and with the output as the variable y. The notation $y \leftarrow B^{C(\cdot)}(x)$ indicates that B can also make repeated queries to an oracle implementing algorithm C, provide all inputs to C, and receive the output. Sometimes we want to denote that the adversary can provide some oracle inputs while other oracle inputs are pre-programmed: for example, the notation $B^{C(\cdot,z)}$ for an algorithm C with two inputs indicates that B gets to provide the first input to C, while the second input to C will be pre-programmed to z.

Protocols. Throughout the paper, we have protocols and ceremonies. A protocol P for a fixed number of n parties will consist of n probabilistic algorithms $P_1, \ldots, P_n$, one for each party. Each of these algorithms P_i takes two inputs – an incoming message, and a state – and produces two outputs – an outgoing message, and an updated state: $P_i(in, st) \to (out, st')$.

Every party is initialized with a local input, which is treated as its first incoming message, and has an empty state to begin with. Then the parties are able to interact. At some point each party terminates and outputs a local output message.

Honest execution of a protocol. Every protocol is accompanied by an interaction pattern for an honest execution of the protocol. To illustrate, a particular protocol's intended interaction pattern may be that the client runs, then the client's output message is given as the input message to the server, then the server's output message is given as the input message to the client, and so on. The honest interaction pattern for an n party protocol with m messages is a vector $\vec{p} \in \mathbb{Z}_n^{m+1}$; the $\vec{p}_i$ is the index of the party that receives the $(i-1)$th message and sends the ith message.

We use the notation $\langle y_1, \ldots, y_n \rangle \leftarrow P\langle x_1, \ldots, x_n \rangle$ to denote giving each party i local input x_i, then running the honest interaction pattern, with party i receiving local output y_i. Specifically, $\langle y_1, \ldots, y_n \rangle \leftarrow P\langle x_1, \ldots, x_n \rangle$ is short-hand for:

1: **for** i from 1 to n **do**
2: $\quad (\perp, st_i) \leftarrow P_i(x_i, \perp)$ ▷ Initialize party i with their local input
3: **end for**
4: $m_0 \leftarrow \perp$
5: **for** i from 1 to $m+1$ **do**
6: $\quad (m_i, st_{\vec{p}_i}) \leftarrow P_{\vec{p}_i}(m_{i-1}, st_{\vec{p}_i})$ ▷ Pass message m_{i-1} to the next party $\vec{p}_i$

7: **end for**

8: **for** i from 1 to n **do**

9: $y_i \leftarrow P_i(\perp, st_i)$ ▷ Party i generates its local output

10: **end for**

For example, if P is a 2-party protocol with 3 message flows in which the first party is the initiator (and thus has honest interaction pattern $\vec{p} = (1, 2, 1, 2)$, the notation $\langle y_1, y_2 \rangle \leftarrow P\langle x_1, x_2 \rangle$ is short-hand for:

1: $(\perp, st_1) \leftarrow P_1(x_1, \perp)$ ▷ Initialize party 1 with their local input

2: $(\perp, st_2) \leftarrow P_2(x_2, \perp)$ ▷ Initialize party 2 with their local input

3: $(m_1, st_1) \leftarrow P_1(\perp, st_1)$ ▷ Generate the initiator's first outgoing message

4: $(m_2, st_2) \leftarrow P_2(m_1, st_2)$ ▷ Pass message to responder

5: $(m_3, st_1) \leftarrow P_1(m_2, st_1)$ ▷ Pass message to initiator

6: $(\perp, st_2) \leftarrow P_2(m_3, st_2)$ ▷ Pass message to responder

7: $y_1 \leftarrow P_1(\perp, st_1)$ ▷ Party 1 generates its local output

8: $y_2 \leftarrow P_2(\perp, st_2)$ ▷ Party 2 generates its local output

Adversary participation in a protocol. In some cases, an adversary is allowed to interact with parties in the context of executing a protocol. The adversary is a network participant, but does not fully control the network. If the honest interaction pattern of the protocol would have two non-adversary-controlled parties interact directly at some point during the protocol, that remains the case: the adversary does not get to see the messages of that interaction, nor control their delivery. As well, the adversary does get to send messages to non-adversary-controlled parties.

The notation $\langle y_1, \ldots, y_n \rangle \leftarrow P(A)\langle x_1, \ldots, x_n \rangle$ denotes an execution of protocol P with the adversary algorithm A participating in the interaction. Here, x_i can be

specified value (or tuple of values), the symbol $\cdot$, a tuple containing specified values or symbols $\cdot$, or the symbol $*$. The symbol $\cdot$ denotes that the adversary A gets to provide a value in that location for the local input of the party. The symbol $*$ denotes that the adversary acts that party in the protocol, sending and receiving all messages for that party. We consider A to be a stateful algorithm, that is, it has additional memory st_A that is an implicit output of one execution and implicit input to the next execution.

Specifically, $P(A)\langle x_1, \ldots, x_n \rangle$ is short-hand for:

for i from 1 to n, $x_i \neq *$ **do**
  Parse x_i as a tuple $(x_{i1}, \ldots, x_{i\ell})$
  for j from 1 to ℓ **do** ▷ Adversary fills in unspecified $(\cdot)$ arguments
    if $x_{ij} = \cdot$ **then**
      $x_{ij} \leftarrow A()$
    end if
  end for
  $(\perp, st_i) \leftarrow P_i(x_i, \perp)$ ▷ Initialize party i with their local input
end for
$m_0 \leftarrow \perp$
for i from 1 to $m + 1$ **do**
  if $x_{\vec{p}_i} = *$ **then** ▷ If the next party is adversary-controlled
    $m_i \leftarrow A(m_{i-1}, st_A)$ ▷ Pass message m_{i-1} to the adversary
  else
    $(m_i, st_{\vec{p}_i}) \leftarrow P_{\vec{p}_i}(m_{i-1}, st_{\vec{p}_i})$ ▷ Pass message m_{i-1} to next party $\vec{p}_i$
  end if

17: **end for**

18: **for** i from 1 to n, $x_i \neq *$ **do**

19: $y_i \leftarrow P_i(\perp, st_i)$ ▷ Party i generates its local output

20: **end for**

The notation $z \leftarrow A^{P\langle x_1,\ldots,x_n\rangle}(y)$ denotes an algorithm A getting to have repeated, sequential interaction with parties executing protocol P. In other words, this is the short-hand form for an adversary algorithm A being run with input y, and being able to execute $P(A)\langle x_1, \ldots, x_n\rangle$ whenever it wants; the adversary does not receive local outputs of non-adversary-controlled parties.

Chapter 3

System Design

Our solution to the problems outlined in Chapter 1 is called the Attested Drop Protocol . It can be described as three ceremonies: Setup, Attestation, and Dropping. The Setup ceremony is the collection of all the necessary actions taken by the entities prior to the start of the following ceremonies. This is where long term information is established and retrieved.

The Attestation ceremony generates the credential that will be used by the leaker. It consists of four sub-ceremonies: Attestment Protocol One, Delivery, Exfiltration and Attestment Protocol Two. Attestment Protocol One covers the initial communications used to establish that an attested drop protocol is taking place, and facilitates the exchange of information necessary for the ceremony to continue. Delivery describes how the challenge value is going to be delivered. The Exfiltration subceremony covers all of the physical steps taken by the leaker to extract and manage the challenge value it received. Attestment Protocol Two covers the steps taken by the parties after the challenge is received, and how the credential is finally established by the leaker.

The Dropping ceremony details how the credential will be used by the leaker. It consists of two sub-ceremonies, Drop and Verification. Drop describes how someone would take the credential generated in the previous steps, combine it with a message and send this combination to a third party. Verification describes how a third party will check the validity of the attestor's signature on the credential.

3.1 Design Goals

To address the problems from Chapter 1, we need to develop a way to provide anonymous authentication without relying on a PKI. Referring to the motivating scenario outlined in Chapter 1.1, we aim to build a system that protects the whistleblower, without revealing their identity and opening them up to reprisal. To achieve anonymity of the whistleblower, the system should have a means to create anonymous communication channels as well as recognize and plan for known re-identification attacks. To allow for the public to trust the authenticity of a credential, the system should aim to prevent the creation of forged signatures.

Instead of a PKI, we can use pre-existing identities that are tied to a communication medium, such as email. This medium must allow us to send un-solicited messages, which can then be used as challenges. Proof of reception can then be used as a means of validating a digital credential generated by the sending party.

As this system is being proposed for the purpose of the high-risk activity of secret leaking, it is imperative to consider all potential identifying elements in the protocol. These might exist outside of the scope of electronic communications, and instead are part of the human actions taken. Thus any attempt to realize this design must explicitly investigate the actions taken by the human elements of the protocol.

3.2 Entities

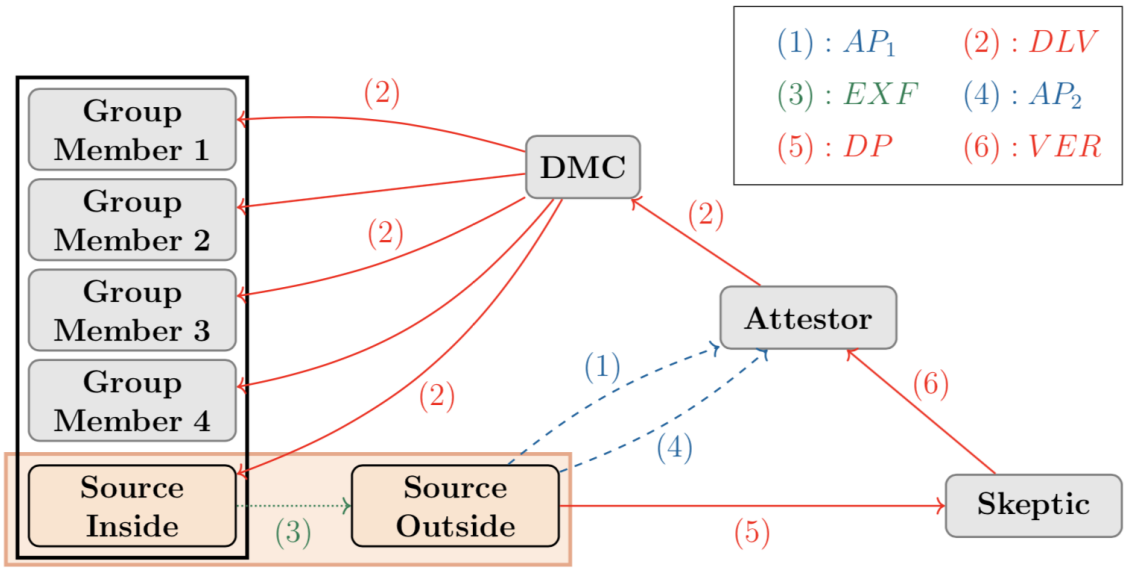

Figure 3.1: Entities of an Attested Drop Protocol

In this figure, red solid lines imply in-band communications, blue dashed lines imply out-of-band communications, and green dotted lines imply that the communication takes place outside of the system

To be able to design a system that achieves these goals, we must first identify and name the various parties that will be a part of it. Figure 3.1 shows the entities and their relations in the system. Each number corresponds to a different sub-ceremony, while the associated arrows describe the direction of the communication in that sub-ceremony.

3.2.1 Source

The **source** acts as a central party to the overall system, and is the one who sends documents. In the motivating example, this would be the whistleblower. He is the one whose identity the system must seek to protect, as he is the one risking repercussions by using the system. The source is represented by a human node in the ceremonies. This way, we can capture how the system could fail because of human error.

The source is assumed to have two devices, **source inside** and **source outside**. The source inside is the source's device within the domain of the organization. It is consider to be monitored by the adversary, so computations related to the protocol should not be done using this device. Source outside is the source's personal device that exists outside of the control of the adversary. This is where the source will send information from, and where they will carry out the computations and cryptographic operations necessary to complete the protocol.

3.2.2 Skeptic

The **skeptic** is the eventual recipient for the leaked documents, and the party for whom the generated credential must convince of its legitimacy. In the motivating example, they are the general public that the whistleblower is planning to show the documents to. In other scenarios they could potentially be another journalist for a news agency, a federal prosecutor, or other members of the organization.

3.2.3 Other Group Members

The **other group members** are the members of the organization who were selected by the source to form the anonymity set of potential leakers for the credential. In the

motivating example, these would be other members of the government organization that would also have access to the financial documents. The other group members each have a public identity tied to the organization, and this identity must itself be tied to the chosen communication medium. The other group members do not have to be aware that the system exists to participate, as they do not have to send any messages only receive them.

3.2.4 Attestor

The **attestor** is an organization that attests that a particular credential belongs to a identity associated with an organization. In the motivating example, they are represented by the reporter from the news organization. In the attested drop protocol the attestor is represented by a computer node, whose purpose is to receive and send messages associated with the system. The attestor is required to have a long-term public key, to allow it to generate publicly verifiable credentials associated with that key.

3.2.5 Delivery Medium

Public messages between parties are sent via a communication mechanism, which will be referred to as the **delivery medium**. In the motivating example, this would be email. Other potential delivery mediums could be instant messaging, phone calls or text messages.

The delivery medium will have publicly known identities associated with it, such that any party would be able to contact any other using the medium. We will assume that all of the identities associated with each of the parties will already exist prior to

the start of the attested drop protocol.

3.2.6 Delivery Medium Controller

The **DMC**, or **delivery medium controller**, is the entity that has control over the delivery medium before it reaches the source inside and the other group members. It is considered to be part of the organization from which the leak is occurring. In the motivating example, this would be the email server controlled by the government organization.

The DMC can be viewed as the computer node that distributes the communications to the various group members, like an email server. The DMC must also be willing to facilitate unprompted communications from outside sources, otherwise the attestor will not be able to send the group messages.

3.3 Ceremonies

A ceremony for a attested drop protocol AD is associated with the following parameters: the number of potential sources n and the namespace consisting of the identities of all possible sources AC. The group of identities selected as potential sources is represented by $G = \{g_1 \dots g_n\}$. The actual source is represented by i, whose identity is $g_i \in G$. It consists of three nodes: **source inside**, the source's computer inside the organization, **source outside**, the source's computer outside of the organization and **source** which represents the human element of the source's behaviour.

The attestor shall be represented by a computer node **attestor**, as it should be a server and requires no human elements. It is sometimes referred to as the **attestation**

server because of this. The messages sent by the attestor are represented by the message list $M = \{m_1, \ldots m_n\}$, where the message directed to the source is m_i. The DMC will be represented by a computer node **DMC**, and the skeptic shall be represented by the computer node **skeptic**.

3.3.1 Setup

The Setup ($SETUP$) ceremony is executed by the attestor and the source outside to initialize the parameters they will use during the ceremonies that follow. It is described by Table 3.1. The attestor must generate the public-private key pair pk_{ATT}, sk_{ATT} to allow for the creation of the credential in AP_1. The source outside needs to retrieve the attestor's public key to start communicating with the attestor in AP_1, if the anonymous communication mechanism the system is using requires it. The exact mechanism for key retrieval is not specified. These computations can occur once, and can be used by many instantiations of the ATT_i and DPG ceremonies if required.

$SETUP$	**Input**	**Output**
Source Outside	$\perp$	$st_{SO}, pk_{AS},$
Attestor	$\perp$	$pk_{AS}, sk_{AS}, st_{AS}$

Table 3.1: Formal Definition of $SETUP$

3.3.2 Attestation

The Attestation (ATT_i) ceremony creates the public verifiable credential that the source can provide with the leaked documents to demonstrate authenticity. The

overall flow of the ceremony can be seen in Figure 3.2. The first sub-ceremony AP_1 represents the initial communications and commitments of the attestor and the source. The DLV and EXF sub-ceremonies describe how the challenge value is sent to and recovered by the source. Finally the AP_2 sub-ceremony covers the actions taken by the source after the challenge is recovered so it can complete the credential.

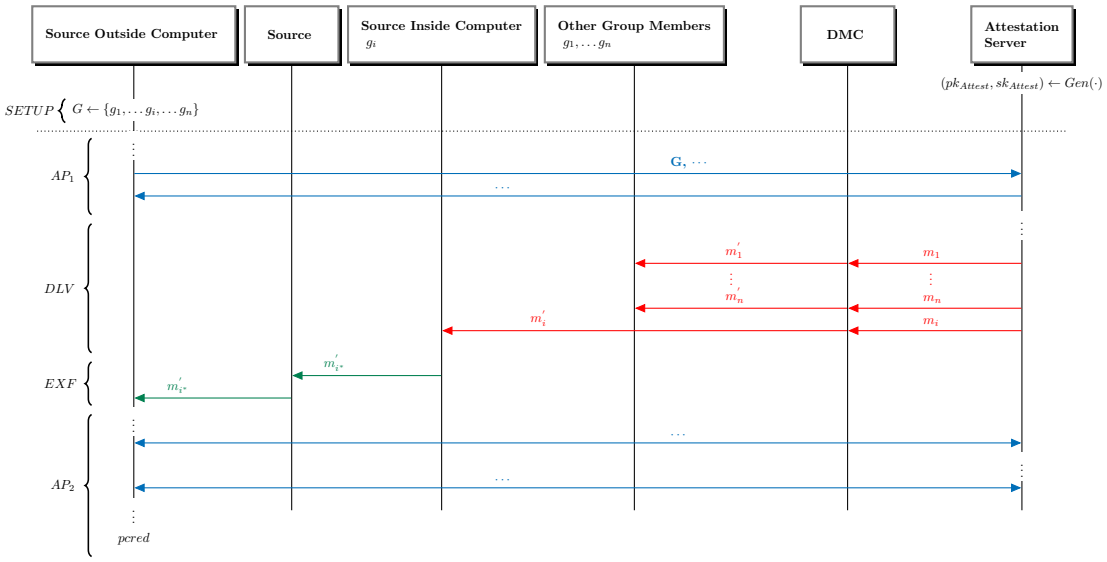

Figure 3.2: Attestment Ceremony Sequence Diagram

3.3.2.1 Attestment Protocol One

Sub-ceremony Attestment Protocol One or AP_1 is the initiation phase of the ceremony ATT_i. It is described by Table 3.2. This contains a message from the source outside to the attestor that must contain a list of potential group members and their communication-medium associated identities. The group list G is selected by the source as input to the ceremony. This message may contain additional information depending on the requirements of the construction.

As the initial communication is completely anonymous, the attestor must decide

if the initial message is credible enough for it to continue with the execution of the ceremony. Otherwise, the attestor could waste its resources creating non-useful credentials. This decision is hard to model, and may require a human component to make proper decisions. In the motivating example, the news reporter could establish out-of-band contact with the source before it decides to generate the credential. Thus we assume that all initial commitments the attestor receives are non-trivial.

The attestor then generates the credential *pcred*. It will also use this credential to generate the messages for DLV, which form the set M. It is possible that the messages consist of the credential, or part of it, signed under a public signing key owned by the attestor.

An instantiation of this ceremony may also include a further communications between the source outside and attestor. Additional information may be required by the attestor from the source outside after the initial commit to generate the credential. The attestor may also send some commitment to the source outside, so the source will have some check against the challenge value it receives.

All communications between entities in this ceremony are considered to be occurring over a secure anonymous channel.

AP_1	**Input**	**Output**
Source Outside	st_{SO}, G, pk_{ATT}	st_{SO}
Attestor	st_{AS}	$st_{AS}, M, G, pcred$

Table 3.2: Formal Definition of AP_1

3.3.2.2 Delivery

Sub-ceremony Delivery or DLV concerns the sending of the challenge from the attestor to the source inside. It is described by Table 3.3. Using the list of identities G provided to it in AP_1 as well as the generated messages M, the attestor will send these messages $m_1^{'} \ldots m_n^{'}$ via the designated communication medium. These are a part of a one-way unsolicited communication, where the receiver is not required to anticipate the message.

The messages will be sent via the DMC, who is in control of the communications into the organization. In practice there may need to be some means of obfuscating the messages so that the DMC is not able to filter the communications and prevent the ceremony from continuing. The DMC could potentially modify the messages in an unpredictable way, but provided the obfuscation is sufficient, the DMC should have no reason to do this. It should be noted that the obfuscation is not considered when ensuring the integrity of the messages, it is just an additional tool used to allow for a practical realization of the protocol.

The source inside and the other group members are the intended recipients of the messages. Presumably unaware that the ceremony is taking place, the other group members should not be able to do anything with the messages they receive. To a non-source group member, these message might appear as a spam message [CL98]. The source inside should recognize the message it receives and prepare for the next sub-ceremony EXF.

DLV	**Input**	**Output**
Attestor	M, st_{AS}	st_{AS}
DMC	$\perp$	M
Source Inside	$\perp$	$m_i^{'}$
Other Group Members	$\perp$	$m_1^{'} \ldots m_n^{'}$

Table 3.3: Formal Definition of DLV

3.3.2.3 Exfiltration

Sub-ceremony Exfiltration or EXF describes how the source moves the challenge value from the message it received $\rightarrow$ $m_i^{'}$ in DLV to its device outside of the organization. It is described by Table 3.4. This ceremony is difficult to model mathematically, as it is completely reliant on human actions. This ceremony still has value, as it can capture the risks and challenges of the human actions.

EXF	**Input**	**Output**
Source Inside	$m_i^{'}$	$\perp$
Source	$\perp$	$\perp$
Source Outside	$\perp$	$m_i^{'}$

Table 3.4: Formal Definition of EXF

3.3.2.4 Attestment Protocol Two

Sub-ceremony Attestment Protocol Two or AP_2 describes how the source outside will complete the recovery of the credential *pcred* from the challenge message. It is

described by Table 3.5. This action may be as simple as de-obfuscating the message or it may involve further communications with the attestor. This ceremony may also contain a step where the source outside checks the challenge against some prior information, like a commitment from the attestor. This helps the source to ensure the value it received was not tampered with.

AP_2	**Input**	**Output**
Source Outside	$st_{SO}, m_i^{'}$	$pcred$
Attestor	st_{AS}	$\perp$

Table 3.5: Formal Definition of AP_2

3.3.3 Dropping

The Dropping or DPG ceremony represents how the source will use the publicly verifiable anonymous credential. The overall flow of the ceremony can be seen in Figure 3.3. This ceremony is named as such because it mimics the idea of a dead-drop, where the source provides a document to a skeptic through a one-way anonymous channel. The skeptic then has no further contact with the source, and has to rely on what was provided in the initial message to authenticate what it received.

The first sub-ceremony DP represents how the source outside authenticates and sends the document and credential to the skeptic. Then, the skeptic can use VER to check the credential against the attestor's public value.

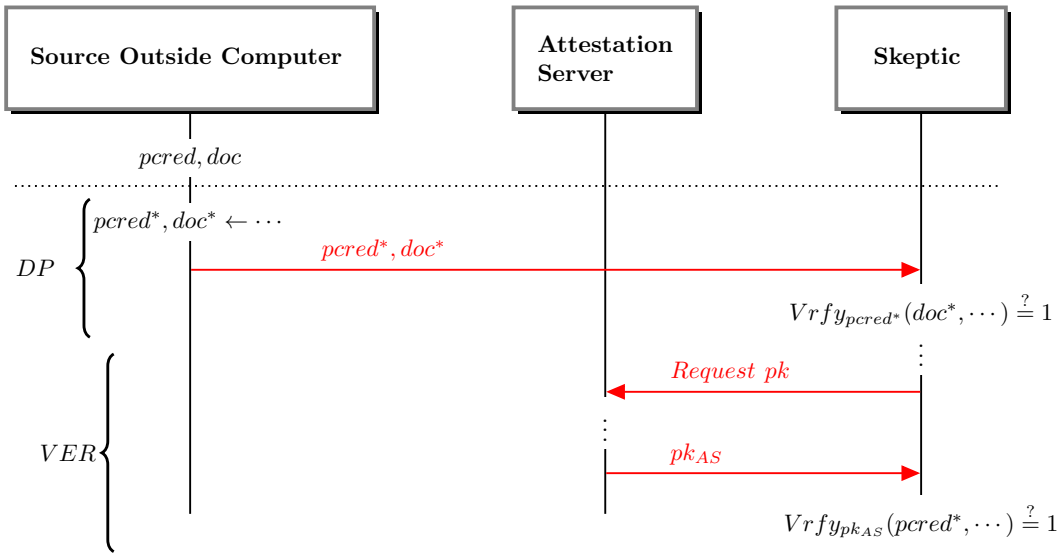

Figure 3.3: Dropping Ceremony Sequence Diagram

3.3.3.1 Drop

Sub-ceremony Drop or DP describes how the source outside uses the credential $pcred$ generated by ATT_i. It is described by Table 3.6. They will need some document or message they intend to give to the skeptic called doc, and this from outside of the system. The source outside will then send the document-credential pair to the skeptic over an anonymous channel. The skeptic may be notified that a message is coming to it from the channel in advance, but this would be done outside of the ceremony. In practice, this may be implemented as a public website or bulletin board that skeptics may access at will.

DP	**Input**	**Output**
Source Outside	$(st_{SO}, sk_{pcred}, pcred, doc)$	$\perp$
Skeptic	$\perp$	$(pcred^*, doc^*, sig^*)$

Table 3.6: Formal Definition of DP

3.3.3.2 Verification

Sub-ceremony Verification or VER describes how the skeptic will check the validity of the credential $pcred^*$ it received in DP with the attestor. It is described by Table 3.7. This will involve acquiring the attestor's public key, as well as any other additional information that may be required in an construction.

VER	**Input**	**Output**
Skeptic	$(pcred^*, doc^*)$	1
Attestor	st_{AS}	$\perp$

Table 3.7: Formal Definition of VER

Chapter 4

Threat Model

In this section, we describe a trust model for the attested drop protocol. First we discuss the assumptions placed on the various entities in the system, based on which parties are honest in a given scenario. Then we state a threat model, formalizing the goals of the system by stating security experiments that any construction could be judged against.

4.1 Trust Model

It is important that before describing threats to a construction, that one be aware of the assumptions placed on the parties. If these assumptions are not clearly stated, a system may be used inappropriately. This results in flaws that could be exploited by an adversary. A detailed trust model will also help with the development of a threat model, as it can be used to emphasize potential weaknesses of the system.

We present this trust model from the perspective of two parties: an honest source and an honest skeptic. Depending on who we are trying to get to trust our system

determines what assumptions we need to place on the other entities. By doing so, we can clearly state what is different in both scenarios.

4.1.1 Source

With an honest source, we assume that the attestor is malicious and trying to learn the identity of the source. A malicious attestor will still try to complete the protocol, but may deviate from their expected behaviours in an attempt to gain information. This may mean that the attestor sends challenge messages that do not correspond to the value associated with the credential.

We also assume that the DMC is malicious in this scenario, but they are not actively filtering out the messages sent from the attestor to the group. They may be doing this because they want to gain information about the identity of the source, and are not concerned with the potential outcomes of a leak.

We also make no assumptions about collusion between parties. If the DMC was in collusion with the attestor, it would not be able to learn anything new that the attestor would not already know.

4.1.2 Skeptic

With an honest skeptic, we assume that the source is acting in good faith and is capable of selecting groups and documents that will not allow an adversary to immediately re-identify them. The system does not make any claims or assumptions about the legitimacy of the documents sent by the source, only that possessing a valid credential will allow the skeptic to verify that these documents came from an identity associated with the organization. It is possible that the source includes some other

evidence as part of the document but that is outside of the scope of the system.

Here we assume that the attestor is honest, and is using the specified hardware and that they have installed all necessary software correctly. Errors in set-up by the attestor can lead to the introduction of side-channel attacks into the system that have not been addressed [Law09]. It is the responsibility of the attestor to make sure their system is well-maintained and regularly updated.

We do assume that the DMC is malicious and is adversarial controlled, but the skeptic does not directly interact with them, so their ability to influence the skeptic is limited.

4.2 Threat Model

Potential threats to users of the system can be broken down into two categories: attacks against the anonymity of the source and attacks against the authenticity of the documents. An attack against the anonymity of the source will seek to re-identify the source after the leak has occurred. An attack against the authenticity of the credential will try to either replicate an existing credential or create a new credential to provide a false sense of authority to the documents.

In this section, we outline the potential attacks against these properties and then provide security experiments to allow an implementer to be able to demonstrate that their construction achieves these goals. A discussion of potential user errors follows each subsection.

4.2.1 Threats Against Privacy (Source Privacy)

An adversary may have the goal of identifying which member of the group is responsible for carrying out the leak. They could have many motivations for wanting to do this, the most concerning of which is retribution. We consider an active adversary for the experiments, who is able to change their expected behaviour in an effort to learn additional information during the execution of protocol.

4.2.1.1 Potential Attacks

A potential way to attack the privacy of the source would be to intercept and monitor the communications between the source and the attestor in the AP_1 and the AP_2 ceremonies.

A global passive adversary would be able to determine who sent the initial communications based on network information [DMS04]. While it is difficult to completely protect against such an adversary, there are anonymous communication systems [DMS04, PHE+17, VDHLZZ15] that any construction of the attested drop protocol should make use of to help prevent monitoring.

If the adversary has control over the messages that are sent by the attestor in DLV, then they could carry out a **splitting attack** on the messages in an attempt to re-identify the source. A splitting attack involves sending different messages to subsets of the group in an attempt to learn information when the source completes the protocol. An adversary could send each group member a message that results in a unique credential, such that the source's identity is revealed to the adversary when they use the credential. If uniqueness is unachievable, the adversary could split the group by sending the same message to only some of the group members, while others receive a

different message. The adversary has to balance how to split the group, where more subsets means more information when a leak occurs while fewer subsets means more information when the leak does not occur.

4.2.1.2 Source Privacy Experiment

The goal of the **Source Privacy Experiment** is to show that an adversary is unable to re-identify the source given all of the information it will receive during the ceremonies. The adversary in this experiment is assumed to control both the DMC and the attestor. After the creation of a credential by the source, the adversary must guess which group member is the source. The adversary is allowed to force the leak of documents under that credential, as well as generate other credentials and leak documents under those. A success in this experiment is defined as the guess being correct.

$\underline{\text{Exp}^{\mathsf{privacy}}_{AD,n}(\mathcal{A})}$

1: $hdl \leftarrow 0$

2: $(pk_{AS}, st_{AS}) \leftarrow SETUP$

3: $G \leftarrow \mathcal{A}^{ATT_*\langle *,*,*,*,*,st_{AS}\rangle, DP\langle *,st_{AS},*\rangle}(n)$

4: $i \xleftarrow{\$} G$

5: $\langle (sk_0, pcred_0), \perp, \perp, \perp, *, *\rangle \leftarrow ATT_i(\mathcal{A})\langle G, \perp, \perp, \perp, *, *\rangle$

6: $i^* \leftarrow \mathcal{A}^{\theta_1, ATT_*\langle *,*,*,*,*,st_{AS}\rangle, DPG\langle *,st_{AS},*\rangle}(n)$

7: **return** 1 if $i^* = i$, else **return** 0

Figure 4.1: Source Privacy Experiment

We define an oracle that will be used by the experiments, called θ_1 and defined in

figure 4.2. θ_1 is an oracle associated with the DPG ceremony. It takes a handle value associated with a public credential i and a message m and generates a signature under that credential on m called σ.

The source privacy experiment only queries oracle θ_1 with $i = 0$, as there should only be one real credential generated. If we allowed unlimited credentials to be issued to the target party, the adversary would be able to use a splitting attack, combined with a binary search to re-identify the source. θ_1 is allowed to accept queries associated with additional handles, as future experiments will be able to make use of this capability.

$\underline{\theta_1(i, m)}$

1: $\langle \perp, st_{AS}, (pcred, doc, \sigma, G) \rangle \leftarrow DPG\langle (sk_i, m), st_{AS}, (pk_{AS}) \rangle$

2: **return** σ

Figure 4.2: Handle Associated DPG Oracle

In an ideal construction, the adversary would have only a $\frac{1}{n}$ chance of being able to re-identify the source. This is the best achievable by the protocol as the adversary will be aware that the source is one of the group members, and is thus able to guess in a uniformly random way at the identity regardless of construction. Thus we are concerned with what gains the adversary can make over this.

$$\mathrm{Adv}^{\mathsf{privacy}}_{AD,n}(\mathcal{A}) = \left| \Pr\left[\mathrm{Exp}^{\mathsf{privacy}}_{AD,n} = 1 \right] - \frac{1}{n} \right|$$

4.2.1.3 User Error

There exists a significant risk of re-identification by the source if the source does not follow the proper procedures for creating anonymous channels outlined by an

construction. It is possible that how and where the source uses the software re-identifies them to a global adversary [WWY+12].

The selection of G by the source is done outside of the scope of ATT_i, but if done incorrectly it could risk providing the adversary with identifying information. If the group chosen is not appropriate, where some if not all of the other group members are not realistic potential sources, then the adversary will be able to reduce the size of group when they carry out re-identification attacks. The source should look for other plausible potential leakers, who have similar privileges and accesses. The group should be selected to be as large as possible, as the relative level of privacy is derived from the size of the group.

The source should also be aware of what information about their identity leaks when they select the document for the protocol. If there is any identifying information attached to the document, or if the source is the only proposed group member who would have access to it, then the source should select another document to leak.

It is possible that the source unknowingly reveals their identity to the adversary when carrying out the EXF sub-ceremony. This sub-ceremony is described as part of the ATT_i ceremony, but it is not used in any of the described security experiments. It is entirely possible that a source could follow the prescribed guidelines for a attested drop protocol, but still re-identify themselves if they fail to extract the challenge discretely.

That is also why it is important for a designer of a attested drop protocol to be aware of the size and value of the challenge they are proposing. If the challenge value is too large to successfully extract in the given scenario, then that construction may not be useful. If an adversary is monitoring via software how long a message is open,

then the challenge value needs to be able to be memorized quickly by the source. Another adversary may be monitoring the work stations of all users with cameras, so the challenge will need to be able to be concealed in an otherwise standard message.

4.2.2 Threats Against Security (Unforgeability)

An adversary may intend to impersonate someone in the organization or send falsely authenticated documents. The adversary may also send false or conflicting documents under a previously generated credential. The credentials created by this system must then be difficult to create externally, while also having a means of linking the credential to the group member who created it initially.

4.2.2.1 Potential Attacks

Outside of stealing the credential directly from the source, an adversary may try to forge a credential to authenticate their own documents. As an outsider to the organization, the adversary would not have access to an identity that could be a valid group member. They could however pose as the source during the anonymous communication phases, and attempt to complete the credential without having access to the challenge sent by the attestor.

An adversary with an identity associated with organization could run the protocol on its own and generate a valid credential. This is not an attack, but a potential misuse of the system. An insider adversary could however try to make use of an existing credential generated by another organization member who has already leaked a document. They would have to forge the final signature on a leaked document without having knowledge of secret information held by the attestor and the source.

An outsider could also attempt this attack, but they would have less information as they would not be able to see the communications from the DMC to the group.

4.2.2.2 Experiments

The goal of the insider unforgeability experiment is to model when an insider to the organization other than the source is able to reuse a credential. This insider may be another group member or the DMC itself. The goal of the adversary is to create a document doc^* that along with a signature σ^*, validates in VER but have never been used together to leak a document through legitimate means.

$\underline{\text{Exp}^{\mathsf{inforge}}_{AD,n}(\mathcal{A})}$

1: $hdl \leftarrow 0$

2: $(pk_{AS}, st_{AS}) \leftarrow SETUP$

3: $(G, i) \leftarrow \mathcal{A}(n)$

4: $\mathcal{A}^{\theta_1,\theta_2,ATT_*\langle *,*,*,*,*,st_{AS}\rangle,DPG\langle *,st_{AS},*\rangle}(n)$

5: $j \leftarrow hdl$

$\langle (sk_j, pcred_j), \perp, \perp, *, *, st_{AS}\rangle \leftarrow ATT_*(\mathcal{A})\langle *, *, *, *, *, st_{AS}\rangle$

$hdl \mathrel{+}= 1$

6: $\mathcal{A}^{\theta_1,\theta_2,ATT_*\langle *,*,*,*,*,st_{AS}\rangle,DPG\langle *,st_{AS},*\rangle}(n)$

7: $\langle *, st_{AS}, (pcred^*, doc^*, sig^*, G^*)\rangle \leftarrow DPG(\mathcal{A})\langle *, st_{AS}, pk_{AS}\rangle$

8: **return** 1 if For j, the hdl of the challenge credential : $pcred^* = pcred_j \wedge$ (j, doc^*) not queried to θ_2, else **return** 0

Figure 4.3: The Insider Unforgeability Experiment

This experiment uses an oracle θ_2 to generate credentials from the challenge group

and source. With this oracle, we can prevent an adversary from trivially winning the experiment by providing the challenge group and source to the ATT_i oracle it already has access to. This oracle is described in figure 4.4.

$\underline{\theta_2()}$

1: $\langle (sk_{hdl}, pcred_{hdl}), \perp, \perp, \perp, *, * \rangle \leftarrow ATT_i(\mathcal{A})\langle G, \perp, \perp, \perp, *, * \rangle$

2: $hdl \mathrel{+}= 1$

3: **return** $pcred_{hdl}, hdl$

Figure 4.4: Fixed Group ATT_i Oracle

$\text{Exp}^{\mathsf{inforge}}_{AD,n}$ starts by initializing hdl and the other initial parameters including the state of the attestor. The adversary selects a group G and the source. It can then run ATT_*, DPG, θ_1 and θ_2, where all queries involving group G and source i will be run through the oracles. The challenge credential $pcred_j$ is then generated, with the associated handle value j. The adversary can then make unlimited queries until it is required to return $pcred^*$, its forged credential that validates under the attestor's public key.

The adversary wins $\text{Exp}^{\mathsf{inforge}}_{AD,n}$ if it can generate a signature on a document that validates under $pcred_j$ but was not queried to θ_1.

The potential advantage gained by the adversary during this experiment should be negligible. In an ideal world, an adversary would not be able decrypt the message in polynomial time.

$$\text{Adv}^{\mathsf{inforge}}_{AD,n}(\mathcal{A}) = \Pr\left[\text{Exp}^{\mathsf{inforge}}_{AD,n} = 1\right] \leq \mathsf{negl}(\lambda)$$

Threats to the authenticity of the credential are not just limited to organizational

insiders. Even if an adversary does not have access to the communications sent across the DMC, they could still try to forge a document and signature. Thus the goal of the outsider unforgeability experiment is to demonstrate that an outsider to the organization is not able reuse an existing credential to create new signatures or generate valid signatures for documents without having completed the attested drop protocol.

The outsider unforgability experiment is defined by Figure 4.5. We allow the adversary access to an ATT_i oracle, as well as a DPG oracle. These allow an adversary to generate credentials, and leak documents signed under those credentials for an arbitrary group. We also allow the adversary access to oracles for just the AP_1 and AP_2 ceremonies. These allow the adversary to simulate scenario where they can execute the phases in the ATT_i ceremony without having access to the communications in the DLV and EXF sub-ceremonies. In these scenarios, the adversary can try to use the information they gain from these sub-ceremonies.

$\underline{\text{Exp}^{\mathsf{outforge}}_{AD,n}(\mathcal{A})}$

1: $hdl \leftarrow 0$

2: $(pk_{AS}, st_{AS}) \leftarrow SETUP$

3: $\mathcal{A}^{\theta_2,\theta_3,ATT_*\langle *,*,*,*,*,st_{AS}\rangle,DPG\langle *,st_{AS},*\rangle,AP_1\langle *,(st_{AS},sk_{AS})\rangle,AP_2\langle *,st_{AS}\rangle}(n)$

4: $\langle *, st_{AS}, (pcred^*, doc^*, sig^*, G^*)\rangle \leftarrow DPG(\mathcal{A})\langle *, st_{AS}, pk_{AS}\rangle$

5: **return** 1 if $[[pcred^* \neq \perp] \wedge [\forall i \leq hdl : pcred^* \neq pcred_i] \wedge G^*$ not queried to $ATT_*()]$
$\vee [\exists i : pcred^* = pcred_i \wedge (i, doc^*)$ not queried to $\theta_2]$,
else **return** 0

Figure 4.5: The Outsider Unforgeability Experiment

The outsider unforgeability experiment makes use of the θ_2 oracle, as defined in Figure 4.4, but the θ_3 oracle defined in Figure 4.6 as well. θ_3 is an oracle for the ATT_i ceremony. Here the adversary is able to provide the group and potential source to the oracle, in contrast to θ_1.

$\underline{\theta_3(G, i)}$

1: $\langle (sk_{hdl}, pcred_{hdl}), \perp, \perp, \perp, *, * \rangle \leftarrow ATT_i(\mathcal{A})\langle G, \perp, \perp, \perp, *, * \rangle$

2: $hdl \mathrel{+}= 1$

3: **return** $pcred_{hdl}, hdl$

Figure 4.6: Chosen Group ATT_i Oracle

As with Insider Unforgeability, the potential advantage gained by the adversary during this experiment should be negligible in respect to the parameter n.

$$\mathrm{Adv}^{\mathsf{outforge}}_{AD,n}(\mathcal{A}) = \Pr\left[\mathrm{Exp}^{\mathsf{outforge}}_{AD,n} = 1\right] \leq \mathsf{negl}(\lambda)$$

4.2.2.3 User Error

The source should also have a secure way of storing the secret aspects of the credential on the device they are planing to use to leak documents from. The source should also avoid sharing their secret keys with other parties.

4.3 What This Does Not Protect Against

The design outlined in Chapter 3 aims to allow its constructions to meet these security and privacy goals, but there are potential threats to source privacy and unforgeability that this protocol does not natively protect against.

4.3.1 Source Impersonation By Attestor

Given that the source's identity is anonymous from all other entities, and that the authentication of the credential comes from the attestment provided by the attestor, it is possible that a malicious attestor impersonates a source and just claims to have run the protocol.

While the design is unable to prevent this type of attack, there exists means to mitigate the threat. If desired, the source can execute the protocol with another attestor using the same signing credential. An skeptic would then understand that both attestors would need to be colluding in order to generate a false leak. This process could be bootstrapped to many attestors, if the skeptic require it.

4.3.2 False Documents

The authentication provided by the attestor for the credential does not cover the content of the documents themselves. This means that a source could run the attested drop protocol to generate a credential and then use it in conjunction with fake documents to try to convince a skeptic of a falsehood. We leave it to the skeptic's judgement to determine the validity of the contents.

Chapter 5

Construction One

This construction is designed to meet the goals previous defined in Chapter 3 and Chapter 4, while only using standard cryptographic primitives. It was also designed with the aim of having as few messages between parties as possible, although this property was not proven. Other notions of performance and usability were not explicitly considered in this design.

In the design of an attested drop protocol construction, selecting the type of credential used is of great importance. It determines what the messages sent in the DLV ceremony will contain.

A standard primitive that meets the cryptographic goals of a credential would be a public key associated with a digital signature scheme, as demonstrating that one could sign under that key would be a sufficient means of proving ownership. However, this credential would not contain any information regarding the group used to generate it. Without any group information, there is nothing tying the public key to the executing of the attested drop protocol. Thus any adversary could generate their own public-private key pair and claim that it came from the protocol. If the protocol instead

uses a certificate containing the public key and the group as the credential then this problem can be avoided.

When instantiating an attested drop protocol, the designer must also consider how many additional messages must be sent between the source and the attestor. To allow the source a means to validate the challenge they received from the DMC, an additional message will be sent from the attestor to the source in AP_1. In this construction, the message will contain the hashed value of the challenge, which is the contents of the message the attestor sends across the public channel.

We must also consider how to protect the messages that are sent across public channels from eavesdropping. To obfuscate the challenges in the DLV sub-ceremony, a steganographic approach is taken for this construction. It is up to the attestor to generate the shared secret value for the symmetric steganography that will be used, and they must share it with the source. This value will be generated using a known steganographic algorithm.

5.1 Specification

When an implementation of a construction is designed, the digital signature scheme to be used as well as the specific steganographic scheme and hash algorithm are specified.

5.1.1 $SETUP$

In the $SETUP$ ceremony, the Attestor will generate a public-private signing key pair using $(pk_{AS}, sk_{AS}) \leftarrow \mathsf{Gen}$. The source will acquire the public certificate, containing

pk_{AS}, through an out of band channel. This channel is not captured by the ceremony, and will not be included in the analysis, as it should not introduce any additional risks to the system. The source will also select the other group members and form the group of potential leakers G, using an arbitrary means to do so.

5.1.2 ATT_i

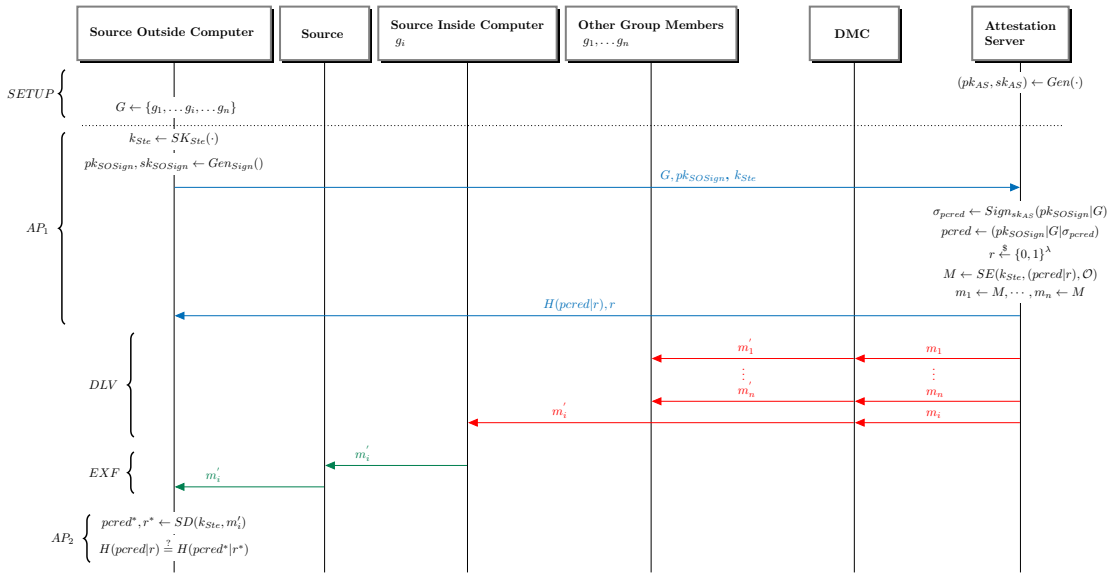

Figure 5.1: Attestment Ceremony Sequence Diagram for Construction One

The Attestation ceremony is specified by four sub-ceremonies: Attestment Protocol One, Delivery, Exfiltration and Attestment Protocol Two. Figure 5.1 shows a sequence diagram describing the various messages sent between the entities in this construction.

5.1.2.1 AP_1

The source outside will generate a signing key pair $(pk_{SOSign}, sk_{SOSign})$ using the digital signature scheme's key generation algorithm Gen. They will also generate the shared secret value k_{Ste} for the steganographic algorithm using its key generation algorithm $\mathsf{SK}_{Steg}()$. The source will open an anonymous communication channel with the attestor and send to them their public signing key pk_{SOSign}, the steganographic secret value k_{Ste} and the list of group members and their associated identities G which they have received as input.

Upon receiving this communication, the attestor will concatenate pk_{SOSign} and G with any additional relevant information to form a certificate. This will be signed using the signing algorithm of the digital signature scheme Sign to create the signature σ_{pcred}. This signature is then combined with pk_{SOSign} and G to create the certificate $pcred$. A random nonce r is then generated. The attestor will then apply the designated steganographic algorithm to the concatenation of the credential and nonce, so it can generate the message set M. Depending on the steganographic algorithm, these messages may be identical or they may be unique for each user. The attestor may also apply further modifications to make each message $m_1 \dots m_n$ unique, if required. Finally the attestor applies the previously designated cryptographic hash algorithm H to $pcred$ to create $\mathsf{H}(pcred)$. The hashed value is sent over the anonymous communication channel to the source outside, so this must be done while the channel is still open.

5.1.2.2 *DLV*

The attestor will now send the messages $m_1, \ldots, m_n$ to each of the identities for the group members $g_1, \ldots g_n \in G$ through the communication medium. The DMC will receive the messages, and perform any modifications it intends to do, creating $m'_1, \ldots, m'_n$. This modifications should not change the content of the messages. It will then forward the messages to the correct addresses.

DLV should only be executed once, to prevent an adversarial attestor from carrying out splitting attacks in an attempt to re-identify the source. Since it is possible that the first message could be dropped for reasons outside of the protocol, an honest attestor may want to resend the challenges to ensure they are received.

5.1.2.3 *EXF*

Now that the source inside has received the message m'_i, it must extract this value to the device source outside. Exactly how the source does this is not specified, but it is important that they do so in such a way that prevents identifying themselves as the source. For example, if m'_i was a binary file, they could save the file to a external memory device and remove that device from the organization.

5.1.2.4 AP_2

Now that the source outside has obtained the value of m'_i, the source can use k_{Ste} and the designated extraction procedure $\mathsf{SD}(k_{Ste}, m'_i)$ to recover the credential $pcred^*$ and the nonce r^*. To verify that this message has not been tampered with, the source can apply the cryptographic hash algorithm to $pcred^*$ concatenated with r^* and compare it to the value it had received in AP_1, $\mathsf{H}(pcred|r)$.

5.1.3 *DPG*

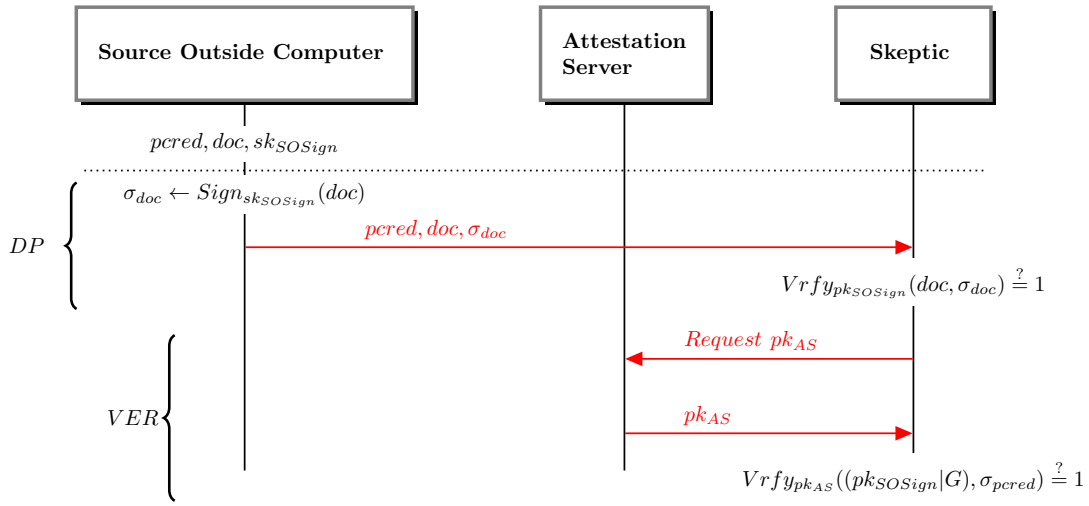

Figure 5.2: Dropping Ceremony Sequence Diagram for Construction One

The Dropping ceremony is specified by two sub-ceremonies: Drop and Verification. Figure 5.2 shows a sequence diagram describing the various messages sent between the entities in this construction.

5.1.3.1 *DP*

The source outside will take as input the public credential *pcred* which was generated by ATT_i, *doc* the document to be leaked, and sk_{SOSign} the secret signing key associated with *pcred*. The source should already have *pcred* and sk_{SOSign} for ATT_i. The source outside will now sign the document using the digital signature algorithm $\mathsf{Sign}_{sk_{SOSign}}(m)$ associated with sk_{SOSign} to create σ_{doc}.

The source will now open an anonymous communication channel with the skeptic, and send to them the document *doc*, the signature on the document σ_{doc} and the credential *pcred*. There may have been communications from the source to the skeptic to alert

them to the incoming message, such as an anonymous email. These are not required by the system and is out of scope of the protocol.

The skeptic can now verify that the signature on the document matches the public signing key that is part of *pcred*, by running the verification algorithm associated with the designated digital signature scheme Ver. If these values match, the skeptic will now want to validate the credential itself.

5.1.3.2 VER

Now in possession of the doc_{signed} and *pcred*, the skeptic will want to verify *pcred*. To do this, they will require the public certificate of the attestor. Thus if the skeptic does not already have this value, it can directly request this value from the attestor. The skeptic can then extract the G, pk_{SOSign} and σ_{pcred} from *pcred*. It can then apply the associated verification algorithm $\mathsf{Ver}_{pk_{AS}}(pk_{SOSign}|G), \sigma_{pcred})$ and see if it is correct. This can be done multiple times if required.

5.2 Security

In Chapter 4, security properties of Source Privacy and Unforgeability were introduced. With this construction we must show that it meets the goals described using the techniques of provable security. We first show that this construction achieves Source Privacy by demonstrating that if the hash function used for creating the hashed-challenge value is collision-resistant, then the adversary has at best a $\frac{2}{n}$ chance of guessing the source's identity correctly. We then show that if the digital signature scheme used to generate signatures in this construction is secure, then the signatures made by the generated credentials will not be able to be forged by a polynomial-time

adversary.

5.2.1 Source Privacy

An obvious starting point for an adversary attempting to learn the identity of the source would be to monitor the communications it sends to the attestor. We assume that the source outside will use an anonymous communication channel to communicate with the attestor, which will avoid these problems.

The adversary will not be able to learn anything about the identity of the source during the dropping ceremony, if the source has not deviated from the prescribed actions until then. This is because all actions taken by the source in DPG can be done anonymously. As such, the adversary is unable to influence the execution of the DPG ceremony and they cannot gain information about the source's identity.

This leaves the delivery sub-ceremony DLV as the only means the adversary has to interact with the system that could reveal information about the source. As the goal of the adversary is re-identify the source, they are not incentivised to make all the messages they send values that could result in source failing to continue the protocol. If no potential source can complete the protocol, they will not learn any information when the ceremony is aborted.

A first attack would then be to make unique challenges for each potential source and wait for the source to leak a document. Then based on which credential is used, the adversary would learn their identity. To prevent this attack, the system specifies that the attestor send a hash of the credential to the source as part of AP_1. Thus as long as the hash function used is collision-resistant, the attestor cannot send different challenges to each group member as they would increase their chances of causing the

protocol to fail. This is because the source will be able to compare what they have received against the challenge and if they are different, they can stop the execution of the protocol.

Theorem 1. *If the underlying hash function* Π *used in the construction is collision-resistant, then an adversary can re-identify the Source with at most* $\frac{2}{n}$ *chance, where* n *is the size of the potential leaking group* G*. More specifically, for each adversary* $\mathcal{A}$ *against the source privacy experiment* $\text{Exp}^{\mathsf{privacy}}_{Imp_1,n}$*, there exists an efficient algorithm* $\mathcal{B}$ *that uses* $\mathcal{A}$ *as a black-box subroutine such that*

$$\text{Adv}^{\mathsf{privacy}}_{Imp_1,n}(\mathcal{A}) \leq n^2 \text{Adv}^{\mathsf{coll}}_{\Pi}(\mathcal{B}) + \frac{2}{n}$$

Proof. For this proof, the main idea is that an adversary cannot do better than randomly guessing the source's identity if it delivers the same challenge to the different partitions of users. In AP_1 the attestor sends a hashed commitment to the source, which will be a binding commitment if the hash function used is collision-resistant. At the time of commitment, the adversary has no information about the potential identity of the user, since all communication up to this stage happens over assumed-anonymous channels. Once the commitment has been received, any attempt by the adversary to deliver distinct messages to different partitions will result in some users receiving challenges that do not match the commitment, thereby allowing the source to detect the adversary's misbehaviour.

The proof is organized through a sequence of games [Sho04]. The first game, G_0, is the original source privacy experiment $\text{Exp}^{\mathsf{privacy}}_{Imp_1,n}$ as stated in sub-subsection 4.2.1.2. The second game G_1 is similar to G_0 but it does not allow for there to be hash function collisions. We can say that G_1 and G_0 are indistinguishable if the hash function is

collision resistant. Finally, we analyse the adversary's win probability in the game G_1, and show that no adversary can do better than a guessing strategy which works with probability at most $\frac{2}{n}$.

Game G_0. This game is the same as privacy experiment from 4.2.1.2

$$\mathrm{Adv}^{\mathsf{privacy}}_{Imp_1,n}(\mathcal{A}) = \Pr\left[\mathrm{Exp}^{\mathsf{privacy}}_{Imp_1,n}(\mathcal{A}) \Rightarrow 1\right] = \Pr[G_0 \Rightarrow 1] \tag{5.2.1}$$

Let $\mathsf{collEvent}$ be the event that, during any execution of the θ_1 oracle on line 5 of $\mathrm{Exp}^{\mathsf{privacy}}_{Imp_1,n}$, the adversary-controlled attestor outputs a set of n messages $m_1, \ldots, m_n$ among which there are two distinct messages, called (x, x'), which hash to the same value.

Game G_1. In this game, the experiment aborts immediately if $\mathsf{collEvent}$ occurs during any execution of the θ_1 oracle. When this event does not occur, game G_0 and G_1 behave identically. By Shoup's Difference Lemma [Sho04, Lemma 1], we have that

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \Pr[\mathsf{collEvent}] \tag{5.2.2}$$

We now show that $\mathsf{collEvent}$ is unlikely to happen if the hash function used is collision-resistant:

Lemma 1. *There exists an efficient algorithm $\mathcal{B}$ that makes calls to $\mathcal{A}$ as a black-box subroutine and satisfies*

$$|\Pr[\mathsf{collEvent}]| \leq n^2 \mathrm{Adv}^{\mathsf{coll}}_{\Pi}(\mathcal{B})$$

where n is the size of the group.

Proof of Lemma 1. Our reduction $\mathcal{B}$ is simply game G_1 itself, where after the attestation server outputs its n messages, the game checks for any collision, and outputs it if one is found. In each query there are $\binom{n}{2}$ pairs of messages to consider. □

Now we can show that in G_1, an adversary cannot do better than splitting the group in half and guessing within that half.

Lemma 2.

$$\Pr[G_1 \Rightarrow 1|\overline{\mathsf{collEvent}}] \leq \frac{2}{n}$$

Proof of Lemma 2. Remember that in this construction of the attested drop protocol that the attestor will only be able to run DLV once, as stated in sub-subsection 5.1.2.2. This can be reflected in the experiment by restricting the number of queries to θ_1 that the adversary makes to 1.

Assume collEvent does not occur in the execution of game G_1.

Let h be the commitment that source outside received during AP_1 in the single call to θ_1. Let $m'_1, \ldots, m'_n$ be the n messages output by the adversarial delivery medium controller.

If there is no message $m \in \{m'_1, \ldots, m'_n\}$ such that $\mathsf{H}(m) = h$, then no honest source, including our source, would be expected to complete the protocol, and the adversary gains no information about the identity of the source.

Otherwise, there is a unique message $m^* \in \{m'_1, \ldots, m'_n\}$ such that $\mathsf{H}(m^*) = h$, although it may be that that same message has been delivered to multiple group members. (Uniqueness of such a message follows from the assumption that collEvent

does not occur.) In particular, suppose that m^* has been delivered to t group members, where $1 \leq t \leq n$.

At the time of delivery, the adversary has no information about the identity of the source relative to the other group members, so, regardless of the adversary's behaviour, the probability that the source inside received m^* such that $\mathsf{H}(m^*) = h$ is exactly $\frac{t}{n}$.

Suppose the source inside received m^*; call this event $\mathsf{rcvdEvent}$. Then the source completes the ceremony, updates its state with m^*, and continues to use the system, including signing messages in θ_2. Its subsequent behaviour is dependent only on m^*. Since exactly t parties also received m^*, the only information the adversary has gained is that the source is among one of the t parties receiving m^*. Thus,

$$\Pr[G_1 \Rightarrow 1 | \mathsf{rcvdEvent} \wedge \overline{\mathsf{collEvent}}] = \frac{1}{t}$$

And of course

$$\Pr[\mathsf{rcvdEvent} | \overline{\mathsf{collEvent}}] = \frac{t}{n}$$

Suppose the source inside did not receive m^*; this is $\overline{\mathsf{rcvdEvent}}$. Then the source does not complete the ceremony, and the ceremony terminates, which is the only observable outcome to the adversary. Since exactly $n - t$ parties also did not receive m^*, the only information the adversary has gained is that the source is among one of the $n - t$ parties not receive m^*. Thus,

$$\Pr[G_1 \Rightarrow 1 | \overline{\mathsf{rcvdEvent}} \wedge \overline{\mathsf{collEvent}}] = \frac{1}{n-t}$$

And so

$$\Pr[\overline{\mathsf{rcvdEvent}}|\overline{\mathsf{collEvent}}] = \frac{n-t}{n}$$

By the law of total probability, we have

$$\begin{aligned}\Pr[G_1 \Rightarrow 1|\overline{\mathsf{collEvent}}] =& \Pr[\mathsf{rcvdEvent}|\overline{\mathsf{collEvent}}] \cdot \Pr[G_1 \Rightarrow 1|\mathsf{rcvdEvent} \wedge \overline{\mathsf{collEvent}}] \\ &+ \Pr[\overline{\mathsf{rcvdEvent}}|\overline{\mathsf{collEvent}}] \cdot \Pr[G_1 \Rightarrow 1|\overline{\mathsf{rcvdEvent}} \wedge \overline{\mathsf{collEvent}}] \\ =& \frac{t}{n} \cdot \frac{1}{t} + \frac{n-t}{n} \cdot \frac{1}{n-t} \\ =& \frac{2}{n}\end{aligned}$$

□

Combining equations (5.2.1) and (5.2.2), and Lemmas 1 and 2, we get the result:

$$\Pr\left[\mathrm{Exp}^{\mathsf{privacy}}_{Imp_1,n}(\mathcal{A}) \Rightarrow 1\right] \leq n^2 \mathrm{Adv}^{\mathsf{coll}}_{\Pi}(\mathcal{B}) + \frac{2}{n}$$

or equivalently

$$\mathrm{Adv}^{\mathsf{privacy}}_{Imp_1,n}(\mathcal{A}) \leq n^2 \mathrm{Adv}^{\mathsf{coll}}_{\Pi}(\mathcal{B}) + \frac{2}{n}$$

□

The previous proof demonstrates that the construction provides protection to the source at a $\frac{2}{n}$ level, but this is worse than the theoretical best level of $\frac{1}{n}$. We do not think it is possible to achieve $\frac{1}{n}$ with this system unless the messages are in a public way, so that each party will be able to see the messages sent to the other parties. However, making these messages public will lead to problems during the proofs for the unforgeability experiments, as we would lose the ability to claim that only members

of the organization could have received the challenge values.

To formalize this idea, we conjecture that a attested drop protocol construction cannot meet the lower bound if it wants to meet both of the privacy and security goals. This is not a formal proof, as there could be some additional means to achieve this that we are not aware of, or a technique that has not been theorized yet.

Conjecture 1. *If the message is sent in a way that it is not publicly visible that each party received the same message, then the lower bound of the chance that an adversary could re-identify the source is* $\frac{2}{n}$ *and not* $\frac{1}{n}$.

If the source cannot see records of the messages that have been sent to the other group members, then a splitting attack is possible. If messages are sent in a publicly visible way, for example over AM radio, then the source can see what the others receive but so could someone outside of the organization. To prevent a splitting attack while still demonstrating that the credential was completed by someone in the organization, there would have to be a means of seeing other messages sent in your organization without revealing your own identity. We cannot think of a system where an low-privilege user would have these accesses, but we cannot state that they do not exist either.

5.2.2 Insider Unforgeability

We can now investigate the unforgeability of the credential generated by the system against an adversary who is an insider to the organization. A member of the organization can always create a new credential for a previously used group, as the adversary can just run the protocol with the same group except this time they are acting as the source. Thus we do not consider these types of attacks in our analysis.

With insider unforgeability, we are worried about an adversary using their knowledge of the messages that were sent to the other group members to be able to sign under the credential without being the source. To prove that this construction is secure against these types of adversaries we provide a reduction from the $\text{Exp}^{\mathsf{inforge}}_{Imp_1,n}$ to the $\text{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ experiment, showing that this property depends on the security of the digital signature scheme used.

Theorem 2. *This construction of the attested drop protocol is insider unforgeable if the digital signature scheme used by the source to sign is secure. More specifically, for each adversary $\mathcal{A}_{inforge}$ against the insider unforgeability experiment $\text{Exp}^{\mathsf{inforge}}_{Imp_1,n}$, there exists an efficient algorithm $\mathcal{A}^1_S$ that uses $\mathcal{A}_{inforge}$ as a black-box subroutine such that*

$$\text{Adv}^{\mathsf{inforge}}_{Imp_1,n}(\mathcal{A}_{inforge}) \leq \text{Adv}^{\mathsf{Sign\text{-}Forge}}_{\mathsf{DS}}(\mathcal{A}^1_S)$$

Proof. Algorithm $\mathcal{A}^1_S$, as shown by Figure 5.3, acts as a challenger to the $\text{Exp}^{\mathsf{inforge}}_{Imp_1,n}$ experiment and a breaker to the $\text{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ experiment for a chosen digital signature scheme. $\text{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ generates a challenge pk, sk and gives pk to $\mathcal{A}^1_S$ as an input. $\mathcal{A}^1_S$ then runs $\mathcal{A}_{inforge}$, replacing the oracle calls to $\theta_2(m, \sigma)$ with calls to the $\mathsf{Sign}_{sk}(\cdot)$ if the $\mathcal{A}_{inforge}$ is requesting signatures for pk. Then, when $\mathcal{A}_{inforge}$ is able to forge a signature for pk, $\mathcal{A}^1_S$ returns to $\text{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ the values of m^* , where it is the value of (pk_{SOhdl}, G) in $pcred^*$ and σ^* is the signature on that tuple that will validate under $\mathsf{Ver}_{pk}(m, \sigma)$

Since $\mathcal{A}^1_S$ is able to sign using the $\text{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ challenge key when $\mathcal{A}_{inforge}$ requests its challenge value, we know that if $\mathcal{A}_{inforge}$ returns a valid $pcred$ it will be able to create

forgeries for $\mathcal{A}_S^1$ using the signing key for the credential. Thus:

$$\mathrm{Adv}^{\mathsf{inforge}}_{Imp_1,n}(\mathcal{A}_{inforge}) \leq \mathrm{Adv}^{\mathsf{Sign\text{-}Forge}}_{\mathsf{DS}}(\mathcal{A}_S^1)$$

But since we have assumed that breaking the digital signature scheme is hard, we have shown that creating a forged credential will be hard.

Algorithm $\mathcal{A}_S^1$

$\mathcal{A}_S^1$ is given pk the challenge public key, and access to a signing oracle $\mathcal{O}_{pk}(m)$

1: $hdl \leftarrow 0$

2: $Q \leftarrow \perp$

3: Run $\mathcal{A}_{inforge}$. The value of hdl when the adversary requests the challenge value is represented by $chlg$.

4: When $\mathcal{A}_{inforge}$ calls $\theta_1(G, i)$ in steps 4 and 6, answer the query in the following way:

- Increment hdl by 1

- $\{pk_{SOhdl}, sk_{SOhdl}\} \leftarrow \mathsf{Gen}(\cdot)$

- If $hdl \neq chlg$, let $pcred_{hdl} = \mathsf{Sign}_{sk_{AS}}(pk_{SOhdl}|G)$

- If $hdl = chlg$, let $pcred_{hdl} = \mathsf{Sign}_{sk_{AS}}(pk|G)$

- Return $(pcred_{hdl}, hdl)$

5: When $\mathcal{A}_{inforge}$ calls $\theta_2(hdl, m)$ in steps 4 and 6, answer the query in the following way:

- $Q \leftarrow Q \cap m$

- If $hdl \neq chlg$, Run $\theta_2(m, \sigma)$

- If $hdl = chlg$, Use $\mathcal{O}_{pk}(m)$ and Return σ^*

6: Whenever $ATT_*\langle\cdots\rangle$ or $LK\langle\cdots\rangle$ are called in steps 4,5,6 and 7 they will execute as expected.

7: Continue answering oracle queries of $\mathcal{A}_{inforge}$ in as above until it chooses to run step 7 and output a new $pcred^*$, as well as a corresponding doc^* and sig^*

8: If $pcred^* = pcred_{chlg}$, then **return** (m^*, σ^*), where m^* is doc^* and σ^* is sig^*

Figure 5.3: Adversary One for Insider Unforgeability in Construction One

□

5.2.3 Outsider Unforgeability

Finally, we can investigate the unforgeability of signatures generated by a credential from the system against an adversary who is an outsider to the organization. In this scenario, we are concerned with two types of attacks. The first is an adversary who is able to generate a new credential without being a member of the group, and are thus able to impersonate a member of the organization. The adversary could do this in one of two ways: they could either break the digital signature scheme used by the attestor and be able to create false credentials that validate under the attestor's public signing key or they could break the commitment sent by the attestor to the source, and acquire a credential that did not go through the delivery ceremony. The other attack is an adversary who is able to generate new signatures under an existing credential, and is able to impersonate the source.

To prove that the construction is secure against both of these attacks we demonstrate that for both, unforgeability of the signatures generated by the credential can reduce to the security of the digital signature scheme that was used. As well, this reduction is not tight but we argue that what is demonstrated is sufficient to show the security.

Theorem 3. *This construction of the attested drop protocol is outsider unforgeable if the digital signature scheme used by the source to sign is secure and the digital signature scheme used by the attestor to sign is secure. We also require that the hash function used by the attestor can operate as a random oracle. More specifically, for each adversary* $\mathcal{A}_{outforge}$ *against the outsider unforgeability experiment* $\mathrm{Exp}^{\mathsf{outforge}}_{Imp_1,n}$*, there exists two efficient algorithms* $\mathcal{A}^2_S$ *and* $\mathcal{A}^3_S$ *that use* $\mathcal{A}_{outforge}$ *as a black-box*

subroutine such that

$$\mathrm{Adv}^{\mathsf{outforge}}_{Imp_1,n}(\mathcal{A}_{outforge}) \leq \mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^2_S) + q\mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^3_S)$$

where q is the number of queries to θ_1.

Proof. There are two scenarios that we must consider. This first reduction is for the scenario where an outsider adversary is able generate a new credential for an arbitrary group. In doing so, the adversary demonstrates that it is able to forge the signing powers of the attestor or they are able to use the commitment sent to bypass the delivery ceremony. The second reduction is for the scenario where an outsider adversary is able to generate signatures for a pre-existing credential they did not create. To do this, the adversary demonstrates that they can forge the signing powers of the source.

Case 1. Similarly to the proof for source privacy, the proof for the first case is organized through a sequence of games. The first game, G_0, is the original outsider unforgeability experiment $\mathrm{Exp}^{\mathsf{outforge}}_{Imp_1,n}$ as stated in Section 4.2.2.2.

The second game G_1 replaces the hashed commitment sent in AP_1 with the hash of a random value. We will show that G_1 and G_0 are indistinguishable to the adversary unless the adversary queries the random oracle on an unknown value.

Finally, we analyse the adversary's win probability in the game G_1, and show that an adversary would need to be able to forge a signature for the chosen digital signature scheme.

Game G_0. This game is the same as unforgeability experiment described by Figure 4.5. Since we are using the random oracle model for these experiments, all calls to the hash function $\mathsf{H}(i)$ are replaced with calls to the random oracle $RO(i)$.

$$\mathrm{Adv}^{\mathsf{outforge}}_{Imp_1,n}(\mathcal{A}) = \Pr\left[\mathrm{Exp}^{\mathsf{outforge}}_{Imp_1,n}(\mathcal{A}) \Rightarrow 1\right] = \Pr[G_0 \Rightarrow 1] \tag{5.2.3}$$

Game G_1. In this game, instead of hashing the $pcred|G$ combined with a random nonce to form the commitment value, we hash a random value instead. The only way for an adversary to be able to distinguish between the two games would be if they are able to know whether the real value was hashed or if a random value was hashed. To do this, they must query the random oracle $RO(i)$ with the real message, including the nonce $r = \{0,1\}^\lambda$, since the nonce value is only present in the delivered message which the outsider will not receive. Without the real message, the adversary will be reduced to guessing the nonce value.

Thus:

Lemma 3.

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq q_{RO} / \left|(0,1)^\lambda\right| \leq \mathsf{negl}(\lambda)$$

where λ is the security parameter.

Proof. To distinguish between the two scenarios, an outsider adversary will need to determine if the commitment it receives in AP_1 is a hash of the message or a hash of a random value. To do this in the random oracle model, they will need to query the oracle with the message value. Let q_{RO} be the number of queries to the random oracle.

The first change we need to make to The Outsider Unforgeability Experiment outlined in Figure 4.5 is how it handles calls to the AP_1 oracle. Instead of sending a hash of the challenge message, AP_1 should send a hash of a randomly generated value m'. We also need to change how the AP_2 oracle functions, as a regular AP_2 oracle will always reject the correct message as the hash was of a random value. The AP_2 oracle must now know the value of $\mathsf{H}(m')$, and just check that this value was received. An adversary will not be aware that when they run AP_2 that their message is not being checked against the hash of the commitment; from their perspective, everything should run as it did in G_0.

q_{RO} is polynomial in the security parameter, as the number of queries to guess the commitment will grow relative to size of the commitment. The nonce r is exponential in the security parameter, as for each possible credential the adversary must search through all possible nonces. Both terms are thus negligible to the security parameter, and are thus smaller than a negligible function. □

Now that we are in G_1, we can show that the security in this game reduces to the security of the signature forgery experiment for the digital signature scheme used in Construction 1. To do this we will show that there is an efficient algorithm $\mathcal{A}_S^2$ which can break the $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ experiment.

Lemma 4.

$$\Pr[G_1 \Rightarrow 1] \leq \mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}_S^2)$$

where n is the size of the group.

Proof. Algorithm $\mathcal{A}_S^2$, as shown by Figure 5.4, acts as a challenger to the $\mathrm{Exp}^{\mathsf{outforge}}_{Imp_1,\lambda}$ experiment and a breaker to the $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ experiment for a chosen digital signature

scheme. $\text{Exp}^{\text{Sig-Forge}}_{\text{DS},\lambda}$ generates a challenge pk, sk and gives pk to $\mathcal{A}^2_S$ as an input. $\mathcal{A}^2_S$ then runs $\mathcal{A}_{outforge}$, with pk, sk acting as the signing key of the attestor. Whenever $\mathcal{A}_{outforge}$ calls $\theta_1(G, i)$ if ATT_i $\langle \dots \rangle$, $\mathcal{A}^2_S$ uses the $\mathsf{Sign}_{Sk}(\cdot)$ oracle to generate the credential. Then when $\mathcal{A}_{outforge}$ is able to forge a signature for pk, $\mathcal{A}^2_S$ returns to $\text{Exp}^{\text{Sig-Forge}}_{\text{DS},\lambda}$ $m^*$and σ^*, where m^* is the value of (pk_{SOhdl}, G) in $pcred^*$ and σ^* is the signature on that tuple that will validate under $Vrfy_{pk}(m, \sigma)$

Algorithm $\mathcal{A}^2_S$

$\mathcal{A}^2_S$ is given pk the challenge public key, and access to a signing oracle $\mathcal{O}_{pk}(m)$

1: Initialize $hdl = 0$

2: $(pk_{AS}, st_{AS}) \leftarrow \{pk, \perp\}$

3: Now run $\mathcal{A}_{outforge}$.

4: Whenever $\mathcal{A}_{outforge}$ calls $\theta_1()$ in step 3, answer the query in the following way:

- Increment hdl by 1
- $\{pk_{SOhdl}, sk_{SOhdl}\} \leftarrow \mathsf{Gen}(\cdot)$
- $pcred_{hdl} \leftarrow \mathcal{O}_{pk}(pk_{SOhdl}, G)$
- Return $(pcred_{hdl}, hdl)$

5: Whenever $\mathcal{A}_{outforge}$ calls ATT_i $\langle \dots \rangle$ in step 3, answer the query in the following way:

- Increment hdl by 1
- $\{pk_{SOhdl}, sk_{SOhdl}\} \leftarrow \mathsf{Gen}(\cdot)$
- Call $\mathcal{O}_{pk}(pk_{SOhdl}, G)$ to generate a credential signed with the challenge key called $pcred_{hdl}$
- Return $(pcred_{hdl}, hdl)$

6: Whenever θ_2 is called in step 3, run the oracle as expected.

7: Whenever DPG is called in steps 3 and 4, run the algorithm as expected.

8: Whenever the AP_1 oracle is called in step 3, answer the query in the following way:
- Respond to any calls to the Random Oracle $RO(m)$ with $h_m \leftarrow \mathsf{H}(\{0,1\}^\lambda)$ and increment q_{RO} by 1
- Otherwise run as expected

9: Whenever the AP_2 oracle is called in step 3, answer the query in the following way:
- Respond to any calls to the Random Oracle $RO(m)$ with the associated h_m
- Otherwise run as expected

10: Continue answering oracle queries of $\mathcal{A}_{outforge}$ until it reaches step 4 and outputs a new $pcred_*$ that verifies under pk

11: Generate (m^*, σ^*), where m^* is the value of (pk_{SOhdl}, G) in $pcred^*$ and σ^* is the signature on that tuple.

12: **return** (m^*, σ^*)

Figure 5.4: Algorithm Two for Outsider Unforgeability in Construction One

□

By combining the previous statements, we have the following result for the first case:

$$\mathsf{Advantage\ Case\ 1} = \mathsf{negl}(\lambda) + \mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^2_S) \tag{5.2.4}$$

Case 2. The following reduction is for the second case where an outsider adversary is able generate a new signature from an existing credential that they did not create. To do this we will show that there is an efficient algorithm $\mathcal{A}^3_S$ which can break the $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ experiment.

Lemma 5.

$$\Pr[G_1 \Rightarrow 1] \leq q\mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^3_S)$$

where n is the size of the group.

Proof. Algorithm $\mathcal{A}^3_S$, as shown by Figure 5.5, acts as a challenger to the $\mathrm{Exp}^{\mathsf{outforge}}_{Imp_1,n}$experiment and a breaker to the $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ experiment for a chosen digital signature scheme. $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ generates a challenge pk, sk and gives pk to $\mathcal{A}^3_S$ as an input. $\mathcal{A}^3_S$ then runs $\mathcal{A}_{outforge}$, replacing the oracle calls to $\theta_2(m, \sigma)$ with calls to the $\mathsf{Sign}_{sk}(\cdot)$ if the $\mathcal{A}_{outforge}$ is requesting signatures for j, a randomly chosen value representing one of the possible oracle calls. Then when $\mathcal{A}_{outforge}$ is able to forge a signature for pk, $\mathcal{A}^3_S$ returns to $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$ the values of $m*$, the document doc^* that had a forgery generated for it and $\sigma*$, a signature sig^* on that message that will validate under $\mathsf{Ver}_{pk}(m, \sigma)$.

Algorithm $\mathcal{A}_S^3$

$\mathcal{A}_S^3$ is given pk the challenge public key, and access to a signing oracle $\mathcal{O}_{pk}(m)$ as well as q the maximum number of possible queries to θ_2

1: $j \xleftarrow{\$} \{1, \ldots, q\}$

2: $(pk_{AS}, st_{AS}) \leftarrow SETUP$

3: Run $\mathcal{A}_{outforge}$. Whenever $\mathcal{A}_{outforge}$ calls $\theta_1(G, i)$ in step 3, answer the query in the following way:

- Increment hdl by 1
- $\{pk_{SOhdl}, sk_{SOhdl}\} \leftarrow \dot{\mathsf{Gen}}()$
- If $hdl \neq j$, Run $\theta_1(G, i)$ as expected
- If $hdl = j$, let $pcred_{hdl} = \mathsf{Sign}_{sk_{AS}}(pk, G)$
- Return $(pcred_{hdl}, hdl)$

4: Whenever $\mathcal{A}_{outforge}$ calls $\theta_2(i, m)$ in step 3, answer the query in the following way:

- If $hdl \neq j$, Run $\theta_2(i, m)$ as expected
- If $hdl = j$, Use $\mathcal{O}_{pk}(m)$

5: Whenever $ATT_*\langle \cdots \rangle$ is called in step 3, run the algorithm as expected.

6: Whenever DPG is called in steps 3 or 4, run the algorithm as expected.

7: Continue answering oracle queries of $\mathcal{A}_{outforge}$ until it reaches step 4 and outputs a new $pcred_*$, as well as a corresponding doc^* and sig^*

8: If $pcred_* = pcred_j$, then **return** (m^*, σ^*), where m^* is doc^* and σ^* is sig^*

Figure 5.5: Adversary Three for Outsider Unforgeability in Construction One

The algorithm succeeds in breaking the $\mathcal{A}_{outforge}$ experiment with probability $\frac{1}{q}$, as we cannot know which $pcred$ the adversary algorithm will forge the signature for.

Thus we have:

$$\mathsf{Advantage\ Case\ 2} = q\mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^3_S) \tag{5.2.5}$$

□

In the first case, we can see that if $\mathcal{A}_{outforge}$ is able to create a $pcred^*$, then that $pcred$ could be used to generate signature on an arbitrary message to beat $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$. In the second case, we can see that if $\mathcal{A}_{outforge}$ is able to generate a $pcred_*$ that matches $pcred_j$ then the document doc^* and the signature created for it sig^* can be used to beat $\mathrm{Exp}^{\mathsf{Sig\text{-}Forge}}_{\mathsf{DS},\lambda}$. Thus in both scenarios, we are able to show that if we can generate a forged credential, we can break the underlying digital signature scheme that was used. Thus:

$$\mathrm{Adv}^{\mathsf{outforge}}_{Imp_1,n}(\mathcal{A}) = \mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^2_S) + q\mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^3_S) + \mathsf{negl}(\lambda) \tag{5.2.6}$$

But since we assume that breaking the digital signature scheme is hard, we have shown that creating a signature on a document when you are not a member of the potential leaking group is hard. □

5.3 Discussion

With this construction all of the stated design and security goals have been met, but there has been no comprehensive consideration for usability and performance. The source will be required to extract a message large enough to contain cryptographic information discretely, which will be difficult in a hostile environment. In such a scenario, an adversary will be able to monitor the actions of the source for suspicious

activity. Having to download and manage a large binary file may be a cause of suspicion. We can address this problem in another construction.

Chapter 6

Construction Two

The goal of this construction is to address a potentially significant usability problem. The file size of the messages sent by the attestor in DLV could be too large to for the source to extract discreetly. In a surveilled environment where all actions taken by human actors are monitored, even copying the data to a drive could allow the adversary to re-identify the source. Thus we need to develop an construction that is able to work using human extractable challenge.

A human extractable challenge is a message that an individual can quickly memorize but will be able to be hashed with enough information entropy such that finding a collision in polynomial time is non-trivial. There has been significant work into password security and usability, which concludes that a 16+ character password is sufficient for providing these goals [KSK+11, KKM+12]. NIST's recent password guidelines [GNF+17] also support this claim.

In literature, passwords composed of natural language words instead of just characters are commonly referred to as passphrases [KSJS07]. Passphrases can be generated to achieve similar security properties as passwords, by maintaining a similar amount

of information entropy. For example, a three-word passphrase selected from a dictionary of 1024 words with replacement has an information theoretic entropy value of approximately 30 bits, which is almost identical to the entropy of a 5 character password selected from 64 characters [SKK+12].

Definition 4. *A passphrase generation algorithm* $\mathsf{Gen}_{passphrase}(dict, m)$ *is a probabilistic polynomial-time algorithm that consists of:*

- *dict is a dictionary of natural language words, of size d*
- *w is the number of words to be selected for the passphrase*

The output of the passphrase generation algorithm is a string *phrase*, consisting of w words selected with replacement from *dict*. This output is the generated passphrase. The set of all possible passphrases that could be generated by this algorithm is referred to as the dictionary space.

While it has not been shown that a passphrase generated this way is inherently easier to memorize than a password [SKK+12], we believe that a passphrase may be easier to obfuscate in a natural language message than a random character string password.

KDF. Similar to Chapter 5, we allow the source a means to validate the message they received from the DMC by requiring an additional message will be sent from the attestor to the source in AP_1. This message will contain the hashed value of a passphrase, which will also be the contents of the message the attestor sends across typical cryptographic value. If the system uses a hash function such as SHA-3, the adversary to brute force a hash table of possible values given the limited message space for passphrases. Thus for this system, the hash should be generated by a cryptographic hash function that allows for slow execution times. To accomplish this

goal, we will use a key derivation function [PM99, KM17, BDK16, BCS16]. These functions can act like a cryptographic hash function [Kra10] in practice, but they have parameters that allow a user to control how long the hash will take to execute.

Definition 5. *Adapted from [YY05]: A generic password based key derivation function can be written as:*

$$y = \mathsf{F}_\lambda(p, s, c)$$

where

- *p is the password or input string*
- *s is the salt value*
- *c is the iteration count*
- *y is the derived key of length λ or output string*

Key derivation functions have a parameter c, which allows the user to control how quickly the hash is computed. The larger the parameter value, the longer the hashing will take. Both s and c are optional parameters in many password based key derivation functions.

For this construction we replace the calls to the hash function $\mathsf{H}(m)$ with calls to a key derivation function acting as a hash function, which we write as $\mathsf{H}(m, s) \equiv \mathsf{F}(m, s, c)$. For the proofs in this chapter, the value for c is assumed to be sufficiently large. We investigate what this means in practice in Section 6.3.

6.1 Specification

For Construction 2, the major difference from Construction 1 is the commitment sent by the attestation server in AP_1 is not a hash of the credential, but a hash of some passphrase that acts a challenge value. This passphrase allows the source to retrieve the credential from the attestor in AP_2. Both the $SETUP$ and DPG ceremonies are the same as they were in Chapter Construction One, as there is nothing new that the parties need to initialize, as well as no new actions they need to take during a protocol execution.

6.1.1 $SETUP$

In the $SETUP$ ceremony, the attestor will generate a public-private signing key pair (pk_{AS}, sk_{AS}). It will then communicate with a Certificate Authority to acquire a signed certificate for its public key, allowing it to sign under that key. The source will acquire the public certificate, containing pk_{AS} through unspecified means. The source will also select the other group members and form the group of potential leakers G, using an arbitrary means to do so. This ceremony is shown in the figure 6.1.

6.1.2 ATT_i

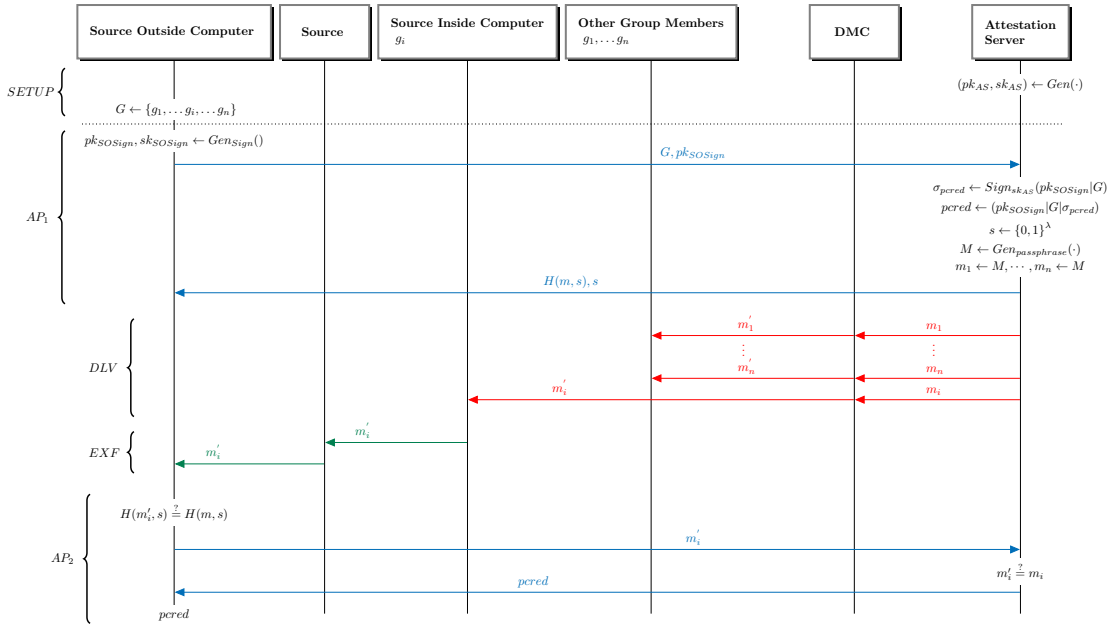

Figure 6.1: Attestment Ceremony Sequence Diagram in Construction Two

The Attestation ceremony specified by four sub-ceremonies: Attestment Protocol One, Delivery, Exfiltration and Attestment Protocol Two. Figure 6.1 shows a sequence diagram describing the various messages sent between the entities in this construction.

6.1.2.1 AP_1

The source outside will generate a signing key pair $(pk_{SOSign}, sk_{SOSign})$ for the digital signature scheme using its key generation algorithm Gen. They will open an anonymous communication channel with the attestor and send to them their public signing key pk_{SOSign}, and the list of group members and their associated identities G which they should have received as input.

Upon receiving this communication, the attestor will concatenate pk_{SOSign} and G with any additional relevant information to form a certificate. This will be signed using the signing algorithm of the chosen digital signature scheme Sign to create the signature σ_{pcred}. This signature is then combined with pk_{SOSign} and G to create the certificate $pcred$. The attestor will now generate the salt for the KDF s, which is a binary value whose length is the security parameter λ. They will then run the passphrase generation algorithm $\mathsf{Gen}_{passphrase}(dict, w)$ to generate the message set M. These messages need to be identical, as the messages will be hashed as the commitment. The attestor may also apply further modifications to make each message $m_1 \dots m_n$ unique, but still contain an identical password.

Finally the attestor uses the KDF to m to create $\mathsf{H}(m, s)$, the hash of m. The hash, along with the salt value, are sent back over the anonymous communication channel to the source outside, and this message must be sent while the channel is still open.

6.1.2.2 *DLV*

The attestor will now send the messages $m_1, \dots, m_n$ to each of the identities for the group members $1, \dots n \in G$ through the communication medium. Assuming it is not aware of the protocol, the DMC will receive the messages perform any modifications it ends to do, creating $m'_1, \dots, m'_n$. It will then forward them to the correct addresses.

6.1.2.3 *EXF*

Now that the source inside has received the message m'_i, it must extract this value to its outside device source outside. Exactly how the source does this is not specified, but it is important that they do so in such a way that prevents identifying themselves

as the source. m'_i should be small enough such that source can quickly memorize the value without needing to physically copy any values.

6.1.2.4 AP_2

Now that the source outside has extracted the value of m'_i, they must first check to see if this message has not been tampered with. To do this, they can apply the cryptographic hash algorithm to m'_i and compare it to the value it had received $H(m'_i)$. If the values match, then the source must again contact the attestor over an anonymous channel, this time sending the recovered message as a password, with the attestor sending the credential *pcred* as a response.

6.1.3 DPG

This ceremony is identical to the ceremony proposed in Chapter 5. This is because the proposed changes from Chapter 5 to Chapter 6 are related to how to the credential is generated and recovered and not how it would be used.

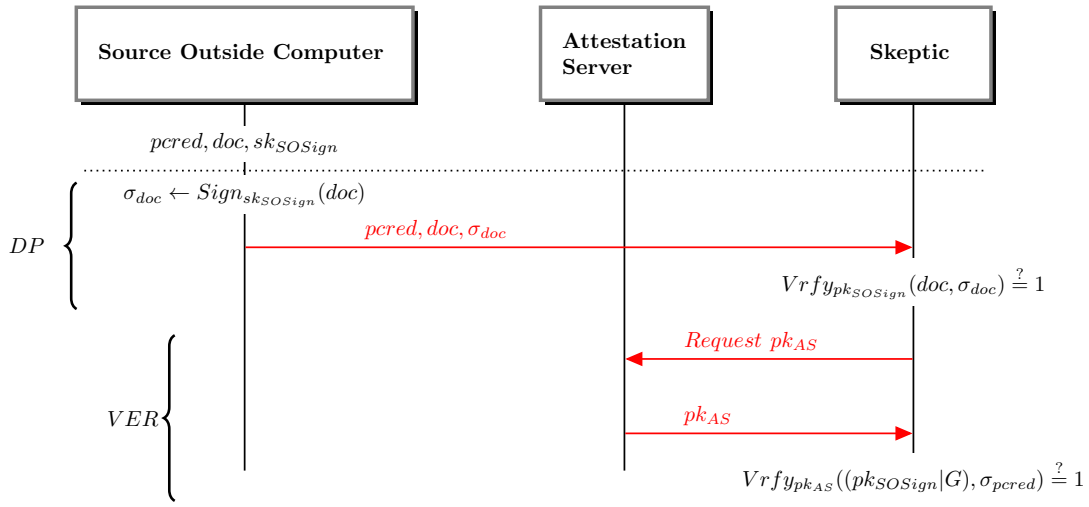

Figure 6.2: Dropping Ceremony Sequence Diagram in Construction Two

The Dropping ceremony is specified by two sub-ceremonies: Drop and Verification. Figure 6.2 shows a sequence diagram describing the various messages sent between the entities in this construction.

6.2 Security

In Chapter 4, security properties of source privacy and unforgeability were introduced. With this construction we must show that it meets those goals using the techniques of provable security.

In comparison to the construction provided in Chapter 5, there are a few more attacks that we must consider. Given that the challenge messages will have a smaller message space than we would prefer, the designer of an implementation based on this construction must set the iteration count and protocol timeouts

6.2.1 Source Privacy

As discussed in subsection 5.2.1, the *DLV* ceremony is where an adversary will interact with the system to gain information about the source. Here the adversary will manipulate the messages sent from the attestor to the group members in an attempt to gain information about the source, by sending unique passphrases to each group member. To prevent this attack, the system specifies that the attestor send a hash of the passphrase to the source as part of AP_1. Thus as long as the key derivation function used is collision-resistant, the attestor cannot send different challenges to each group member.

Given that passphrases inherently have a lower information entropy value than random strings, an adversary could realistically perform an offline dictionary attack on the passphrase space to find collisions. To prevent this, we can make use of the salt parameter in the key derivation function. If the source sends a large enough salt value in its initial communication, the adversary without prior knowledge of the salt will not be able to carry out a dictionary attack.

Theorem 4. *If the underlying key derivation function* F *used in the construction for hashing is collision-resistant, then an adversary can re-identify the source with at most* $\frac{2}{n}$ *chance, where* n *is the size of the potential leaking group* G*. More specifically, for each adversary* $\mathcal{A}$ *against the source privacy experiment* $\mathrm{Exp}^{\mathsf{privacy}}_{Imp_2,n}$*, there exists an efficient algorithm* $\mathcal{B}$ *that uses* $\mathcal{A}$ *as a black-box subroutine such that*

$$\mathrm{Adv}^{\mathsf{privacy}}_{Imp_2,n}(\mathcal{A}) \leq n^2 \mathrm{Adv}^{\mathsf{coll}}_{\mathsf{F}}(\mathcal{B}) + \frac{2}{n}$$

Proof. The proof for this theorem is very similar to the proof for source privacy in

Construction 1. Instead of using a nonce value to concatenate with the challenge value to extend the message space of hashed commitment, this implementation uses a salt parameter for the KDF to achieve the same objective. Thus to have negligible probability for adversary to distinguish between G_0 and G_1, we would need the salt to be sufficient large. For this proof, we will assume that we have selected a large enough value, and we will address the selection of a proper size in Section 6.3.
The remainder of the proof would proceed in an identical fashion to the proof in Subsection 5.2.1. □

6.2.2 Insider Unforgeablity

With insider unforgeability, we are worried about an adversary using their knowledge of the messages that were sent to the other group members to be able to sign under the credential without being the source. These attacks do not change based on the changes between Construction 1 and Construction 2. This is because an insider already has access to the communications sent to the group members in DLV. Thus an insider adversary will have direct access to the passphrase and can recover *pcred* at will.

Theorem 5. *This construction of the attested drop protocol is insider unforgeable if the digital signature scheme used by the source to sign is secure. More specifically, for each adversary $\mathcal{A}_{inforge}$ against the insider unforgeability experiment $\mathrm{Exp}^{\mathsf{inforge}}_{Imp_2,n}$, there exists an efficient algorithm $\mathcal{A}^4_S$ that uses $\mathcal{A}_{inforge}$ as a black-box subroutine such that*

$$\mathrm{Adv}^{\mathsf{inforge}}_{Imp_2,n}(\mathcal{A}_{inforge}) \leq \mathrm{Adv}^{\mathsf{Sign\text{-}Forge}}_{\mathsf{DS}}(\mathcal{A}^4_S)$$

Proof. This proof is identical to the proof from Subsection 5.2.2. □

6.2.3 Outsider Unforgeability

We can also demonstrate that signatures created by credentials from this construction are unable to be forged by an organizational outsider. Similarly to the discussion in Subsection 5.2.3, there are two attack scenarios. In the first, an adversary creates a valid credential without having access to the messages sent to the group members. The other scenario is where an adversary can generate new signatures under an existing attested credential, which they do not already possess.

To prove that this construction is secure in both of these scenarios, we demonstrate that the unforgeability of the signatures generated by the credential can reduce to the security of the digital signature scheme that was used. In this construction, we are also worried about potential brute force attacks on the dictionary space. Thus we must be aware of the parameter values used by this function when determining the security.

Theorem 6. *This construction of the attested drop protocol is outsider unforgeable if the digital signature scheme used by the source to sign is secure, the digital signature scheme used by the attestor to sign is secure and the output space for the passphrase generation algorithm is big enough to make a dictionary attack infeasible. We also require that the hash function used by the attestor is assumed to be a random oracle. More specifically, for each adversary $\mathcal{A}_{outforge}$ against the outsider unforgeability experiment $\text{Exp}^{\mathsf{outforge}}_{Imp_2,n}$, there exists two efficient algorithms $\mathcal{A}^5_S$ and $\mathcal{A}^6_S$ that*

uses $\mathcal{A}_{outforge}$ as a black-box subroutine such that

$$\mathrm{Adv}^{\mathsf{outforge}}_{Imp_2,n}(\mathcal{A}_{outforge}) \leq \mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^5_S) + q\mathrm{Adv}^{\mathsf{Sig\text{-}forge}}_{(\mathsf{DS},n)}(\mathcal{A}^6_S) + q_{RO}/\binom{dict}{w}$$

where q is the number of queries to θ_1, q_{RO} is the number of queries to the random oracle, dict is the dictionary of potential words for the passphrase and w is the number of words in the passphrase.

Proof. As we have seen in the proof for Theorem 3 there are two scenarios that we must consider. This first is for the scenario where an outsider adversary is able generate a credential for an arbitrary group. Here, the adversary is demonstrating that they can forge the signature of the attestor. The second reduction is for the scenario where an outsider adversary is able to generate signatures for a pre-existing credential they did not create. Here the adversary demonstrates that they can forge the signature of the source.

Case 1. As we have seen in previous proofs, this proof is organized through a sequence of games. The first game, G_0, is the original outsider unforgeability experiment $\mathrm{Exp}^{\mathsf{outforge}}_{Imp_2,n}$ as stated in Section 4.2.2.2.

The second game G_1 replaces the hashed commitment sent in AP_1 with the hash of a random value. We will show that G_1 and G_0 are indistinguishable to the adversary unless the adversary queries the random oracle on an unknown value. In this construction, the adversary already has access to the salt value used from AP_1, so we cannot use that as a means of increasing the password entropy space. Instead we rely on the adversary's inability to efficiently compute hashes for the dictionary space.

Then we analyse the adversary's win probability in the game G_1. We show that an

adversary would need to be able to forge a signature for the chosen digital signature scheme to be able to forge a signature for a credential generated by the construction.

Game G_0. This game is set in the same environment as the unforgeability experiment from Figure 4.5. Since we are using the random oracle model for this experiment, all calls to the KDF acting as the hash function $\mathsf{H}(i, s)$ are replaced with calls to the random oracle $RO(i, s)$.

$$\mathrm{Adv}^{\mathsf{outforge}}_{Imp_2,n}(\mathcal{A}) = \Pr\left[\mathrm{Exp}^{\mathsf{outforge}}_{Imp_2,n}(\mathcal{A}) \Rightarrow 1\right] = \Pr[G_0 \Rightarrow 1] \tag{6.2.1}$$

Game G_1. In this game, instead of hashing the passphrase *phrase* and the salt s, the random oracle hashes a random value instead.

The only way for an adversary to be able to distinguish between the two games would be if they are able to know whether the real value was hashed or if a random value was hashed. To do this, they must query the random oracle $RO(i, s)$ with the real message, which as an outsider they will not have.

Thus:

Lemma 6.

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq q_{RO} / \binom{dict}{w}$$

where q_{RO} is the number of queries to the random oracle, dict is the dictionary of potential words for the passphrase and w is the number of words in the passphrase.

Proof. For an outsider adversary to be able to distinguish between the two scenarios, they will need to determine if the commitment it receives in AP_1 is a hash of the message or a hash of a random value. To do this in the random oracle model, they

will need to query the oracle with the message value. Let q_{RO} be the number of queries made by the adversary to the random oracle.

The first change we need to make to The Outsider Unforgeability Experiment outlined in Figure 4.5 is how it handles calls to the AP_1 oracle. Instead of sending a hash of the challenge message, AP_1 should send a hash of a randomly generated value m' and the salt s. We also need to change how the AP_2 oracle functions, as a regular AP_2 oracle will always reject the correct message as the hash was of a random value. The AP_2 oracle must now know the value of $H(m', s)$, and just check that this value was received.

For this construction, the dictionary space of the passphrases will be considerable smaller than the dictionary space of the challenge values in Construction 1. This means that it would be possible for an adversary to efficiently determine the hashed values of messages from the dictionary space given that they have the salt value used. Let $\binom{dict}{w}$ be the size of hte dictionary space for this passphrase generation algorithm. Given enough queries, the adversary will be able to test all possible passphrases in the random oracle, and will then be able to determine if the value they have received is part of that set or was a hash of a random value.

This value is not neglible, so we cannot eliminate it from our reduction. However, we will assume for the rest of the proof that we have limited the adversary's ability to make large amount of queries. A discussion on how we will do this is provided in Section 6.3. □

In G_1, it is possible to show that the unforgeability of signatures generated by credentials in this game reduces to the security of the signature forgery experiment for the digital signature scheme used in Construction 2. To do this we will show that

there is an efficient algorithm $\mathcal{A}_S^5$ which can break the $\text{Exp}_{\mathsf{DS},\lambda}^{\mathsf{Sig\text{-}Forge}}$ experiment.

Lemma 7.

$$\Pr[G_1 \Rightarrow 1] \leq \text{Adv}_{(\mathsf{DS},\lambda)}^{\mathsf{Sig\text{-}forge}}(\mathcal{A}_S^5)$$

where λ is the security parameter.

Proof. This proof is identical to the proof of Lemma 4, except that the implementation the hash function uses a salt parameter instead of the message being concatenated with a nonce. This change does not directly affect the results of the proof as we already use a random oracle to model the hash function. □

Thus by combining the previous statements, we have the following result for the first case:

$$\mathsf{Advantage\ Case\ 1} = q_{RO}/\binom{dict}{w} + \text{Adv}_{(\mathsf{DS},\lambda)}^{\mathsf{Sig\text{-}forge}}(\mathcal{A}_S^5) \tag{6.2.2}$$

Case 2. The following reduction is for the second case where an outsider adversary is able generate a new signature from an existing credential that did not create. To do this we will show that there is an efficient algorithm $\mathcal{A}_S^6$ which can break the $\text{Exp}_{\mathsf{DS},\lambda}^{\mathsf{Sig\text{-}Forge}}$ experiment.

$$\mathsf{Advantage\ Case\ 2} = q\text{Adv}_{(\mathsf{DS},\lambda)}^{\mathsf{Sig\text{-}forge}}(\mathcal{A}_S^6) \tag{6.2.3}$$

Proof. This proof is identical to the proof of 5, except that the implementation the hash function uses a salt parameter instead of the message being concatenated with a nonce. This change does not directly affect the results of the proof as we already use a random oracle to model the hash function.

Thus we have:

$$\textsf{Advantage Case 2} = q\text{Adv}^{\textsf{Sig-forge}}_{(\textsf{DS},\lambda)}(\mathcal{A}^6_S) \tag{6.2.4}$$

□

For the first case, if $\mathcal{A}_{outforge}$ is able to create a $pcred^*$, then that $pcred$ could be used to generate signature on an arbitrary message to beat the $\text{Exp}^{\textsf{Sig-Forge}}_{\textsf{DS},\lambda}$ experiment. For the second case, if $\mathcal{A}_{outforge}$ is able to generate a $pcred_*$ that matches $pcred_j$ then the document doc^* and the signature created for it sig^* can be used to beat the $\text{Exp}^{\textsf{Sig-Forge}}_{\textsf{DS},\lambda}$ experiment. Therefore we are able to show that if we can generate a forged credential, we can break the underlying digital signature scheme that was used. Thus:

$$\text{Adv}^{\textsf{outforge}}_{Imp_2,n}(\mathcal{A}) = \text{Adv}^{\textsf{Sig-forge}}_{(\textsf{DS},\lambda)}(\mathcal{A}^5_S) + q\text{Adv}^{\textsf{Sig-forge}}_{(\textsf{DS},\lambda)}(\mathcal{A}^6_S) + q_{RO}/\binom{dict}{w} \tag{6.2.5}$$

But since we assume that the digital signature scheme is unforgeable, breaking outsider unforgeability is as hard as an offline-dictionary attack against the passphrase. We can prevent this by controlling how quickly the hash function works. To do this, we need to select a large enough c value for the KDF used, in respect to how much time we are allowing for a protocol timeout.

□

6.3 Discussion

Unlike standard cryptographic hash functions, KDFs include optional parameters that the user may set to modify the behaviour of the function. The following is a

discussion about how we set those parameter values in respect to the various security properties we have proven in Section 6.2.

Source Privacy For the source privacy property, the salt value used by the KDF is an important parameter to demonstrate security. Salt values are included to prevent an adversary from precomputing a dictionary of passphrase,hash pairs in advance of the protocol execution. To prevent attacks against the attestation server's commitment used during the attestation ceremony, a salt value should be sent from the source to the attestor during AP_1. The KDF will then use the salt to help generate the messages M. Because the message space of the KDF without the salt is potentially small (2^{30}), an adversary could test offline all possible messages to find a preimage. NIST suggests that the value for the salt be no shorter than 128 bits [TBBC10].

Insider Unforgeability. Given that an insider will have access to the challenge message directly, they do not need try to derive it from the commitment. So, for this property the exact parameter values used by the KDF are not significant for the proof.

Outsider Unforgeability. We have so far ignored the third parameter c in our KDF definition when discussing how we are using it as a cryptographic hash function. The value c is the iteration count. It refers to how many times the hash function is applied repeatedly to the message: $H(H(H \dots H(m) \dots)) = H^c(m)$, where the hash function is applied c times.

In a set of recommendations [TBBC10], NIST recommends that this value should be set as large as possible for the application. It is stated that this value should

be at a minimum 1000, but a value as large as 1000000 may be appropriate. More recent NIST documents suggest that the minimum value should be closer to 10000 [GNF+17].

While there does not appear to be a generic formula designed to estimate the minimum number of iterations for a KDF to achieve security against dictionary attacks, this does not mean that we cannot describe one. Consider the following formula:

$$\frac{1}{2}|\mathcal{P}| \geq \frac{a}{c} * t \tag{6.3.1}$$

Where $|\mathcal{P}|$ is the size of the message space that the passphrase is derived from, c is the number of iterations used by the KDF, a is the adversary power in protocol iterations per second and t is the protocol timeout in seconds. This equation then represents how much greater the average time required to find a hash value in the passphrase search space needs to be compared to the timeout value we set for the protocol.

We can derive appropriate numbers for these values, using some example values. The approximate search space for a human-extractable passphrase is 2^{30} [SKK+12]. A reasonable value for the protocol timeout would be one week, or approximately 2^{19} seconds, as that would allow the attestor to attempt to send the message multiple times. A recent study into the cost of computing PBKDF2 [VMBM19] suggests that standard GPU configurations seen in common commercial desktop computers can compute between approximately 75,000,000 and 191,000,000 iterations per second, if the hash function used is SHA-256. We will approximate these values by saying that an adversary has an attack power a of 2^{28} iterations per second divided by an iteration count of c.

When we substitute these values into the formula we can solve for the iteration count

c:

$$\frac{1}{2}|\mathcal{P}| \geq \frac{a}{c} * t$$
$$\frac{1}{2} * 2^{30} \geq \frac{2^{28}}{c} * 2^{19}$$
$$c \geq 2^{18}$$

This suggest that to protect our construction against dictionary attacks, we would require that KDF use at a minimum of 2^{18}, or about 260,000, iterations for it to be secure. To account for stronger adversaries, who have access to more powerful GPU setups, it would likely be possible to use an iteration count value that is greater than 10,000,000. We can do this for this construction, as the source and attestor should only be hashing a few values during the execution of the protocol. For an honest party using a lower end standard GPU setup, which can compute approximately 75,000,000 iterations per second [VMBM19], they would be able to determine the hash of the commitment in $\frac{10,000,000}{75,000,000}$sec which is approximately 0.13 seconds.

Chapter 7

Conclusion

In Chapters 5 and 6 we described two secret leaking scenarios and present constructions to meet the requirements of those scenarios. Every secret leaking scenario provides its own challenges, so other scenarios may require solutions different than what we have presented. In this chapter, we first review the possible design decisions that can be made when creating an attested drop protocol construction.

There are still technical and theoretical challenges that exist in regards to the design of attested drop protocols that would make for interesting future work. We can also look beyond secret leaking, and see that it is possible to adapt the attested drop protocol to build constructions that address other potential applications such as reviewing systems.

7.1 Design Options

When creating a construction for an attested drop protocol, there are a series of design choices that must me made. The correct choice depends on the intended use

case for the construction, as well as which technologies are available to each party. The following section is a discussion around some of these points and their potential options.

7.1.1 Obfuscated Messages

An attested drop protocol construction may require that the messages sent from the attestor to the source inside and other group members are obfuscated such that the DMC is unable to determine if the messages are related to the protocol's execution. Without obfuscation, a real-world adversary could detect and filter out all of the challenge messages. A construction could obfuscate its messages by hiding the contents of the challenge using encryption, or by hiding the existence of the challenge completely with steganography. It is also possible to use a combination of both approaches.

In Chapter 5, a steganographic algorithm is used. The source first sends to the attestor a value that acts as a shared private key for a previously specified steganographic scheme. The attestor hides the credential value in another plaintext message using the steganographic scheme and sends the output message as the challenge to the group. The second plaintext message could consist of an image, video or binary data file depending on the algorithm that is being used. To the DMC and other group members, the challenge should appear as a normal message from everyday communications provided that the steganographic scheme used is good.

In Chapter 6, we do not directly obfuscate the challenges. The challenge value used in this construction is a passphrase which consists of natural language words. We assume that the attestor is able to hide the passphrase into a natural language message, but we do not specify how. A one-time steganographic scheme could be used to achieve

this if necessary.

One could use a public key encryption scheme to obfuscate the challenge messages. This would not hide that the protocol is being carried out, but it would prevent a polynomial time adversary from determining the contents of the challenge messages. This may be useful in scenarios where the group members regularly receive encrypted messages, or where the DMC actively wants the protocol to succeed. One can imagine a corporation implementing this protocol to handle internal whistleblowing scenarios; the company wants to receive honest feedback, while the whistleblower does not want to reveal their exact identity.

7.1.2 Message Structure

When designing a construction of an attested drop protocol, it is important to specify what the challenge message sent from the attestor to the group members will contain. Otherwise the source will not know what to look for after he has initiated the protocol. First, the designer must decide if the message needs to be human-extractable. This problem is addressed in Chapter 6, where we describe a construction that allows for a human actor to be able to quickly memorize the contents of the challenge so that they can minimize the risk of being exposure.

In scenarios where the challenge messages need only to be computer-extractable, the designer may not have to worry about the challenge message size. With longer challenges, it would be possible to avoid additional, potentially risky communications in sub-ceremony AP_2. In Chapter 5, we describe such a system. Here the message contents are obfuscated, but are not constrained by length. The challenges could be a large binary files. Of course, there would exist a limit to the size of files, as an

attestor cannot send a file that is too large to be delivered.

7.1.3 Message Verifiability

Another design point to consider when creating an attested drop protocol construction is how the system gives the source confidence that the challenge message they received will not provide any additional re-identification risk. Without verifiability, an adversary controlling the attestor could generate multiple credentials, and use those to re-identify the source based on which one was used in the leak. To prevent this, we need the attestor to commit to a value and for the source to have access to this value. We have solved this problem in both of our constructions, but there are other possible solutions.

Hashing. In Chapter 5 and Chapter 6, the design specifies that the attestor send an additional message to the source in AP_1. This message is hashed using a hash function that was selected by the designer. If the hash function used is collison-resistant, the adversary will be unable to efficiently create multiple credentials that hash to the same value.

Bulletin Board. In scenarios where the attestor cannot send additional messages, the designer may use a bulletin board approach. Bulletin boards are frequently seen in e-voting systems [Adi08], where a public list of votes are maintained on a server such that any party can view the results. In general a bulletin board is a public posting of some information indexed by smaller values.

When used as part of an attested drop protocol, the attestor would sent the credential to additional entity which we call the bulletin board. The source could retrieve the

credential by interacting with the bulletin board by using the challenge value they received as the index. To prevent a malicious attestor from re-identifying the source from the index he uses, Private Information Retrieval (PIR) [DGH12] or a similar oblivious transfer mechanism would need to be implemented.

7.1.4 Anonymous Communication Channel Generation

In the AP_1 and AP_2 sub-ceremonies the source creates an anonymous communication channel to contact the attestor. The means by which this channel is created are left up to the designer. Possible factors one can use to determine the appropriate technology include the size of the communications and if any party will need offline access to the messages.

A popular means for generating anonymous channels is the Tor anonymity network [DMS04]. Using this method, the source would need to acquire the address of the attestment server in the Tor network before creating a TCP connection using their onion proxy. As there are over 3 million Tor users as of writing [Tor], Tor can provide the largest anonymity net of all solutions we have investigated.

There are other means of creating this channel besides Tor. Loopix [PHE+17] allows for the source to send the communications as a text message, and allows them to use an email-like interface for off-line access. Vuvuzela [VDHLZZ15] allows the source to send anonymous chat messages which would be protected by a differential privacy mechanism. This mechanism allows for relatively low latency for the communications.

7.1.5 Protocol Timeout

Another decision point for the designer of an attested drop protocol construction is how much time they allow for an attestor to send the challenge to the group members. It is possible that the challenges fail to reach their destination, so an honest attestor may want to resend the challenge. But a dishonest attestor could use multiple attempts to perform splitting attacks on the group, where they send the challenges to the group in waves. In this attack scenario, if the source leaks immediately after receiving the challenge, the attestor will know which wave the source belongs to.

It is then a requirement that the source abort the protocol if it does not receive the challenge after a certain time, and moreover that they do not start AP_2 until a fixed amount time has passed. The attestor will be allowed to resend the message, but the source should only use any future challenges it receives to compare to the initial value it has. If they differ, the source can abort the protocol.

What constitutes a reasonable timeout length depends on the application. For example, in a secret leaking scenario, a timeout of a week will allow for the source to have the time to extract the challenge and for the attestor to have multiple attempts. In a course review scenario, the timeout may only need to be a few hours long.

7.1.6 Multiple Attestors

As discussed in Section 4.3, the design of the attested drop protocol does not protect against scenarios where an attestor pretends to be a source. To provide a skeptic with some assurance that source and attestor are independent, a source can reuse their signing key with a different attestor. This will result in two credentials for the same signing key. A skeptic just needs to believe that one of the attestors is not

colluding with the source, for them to believe in the signing key associated with a credential.

In a secret leaking scenario, the source could execute the protocol with two media organizations of differing political alignments. If the skeptic adheres to the belief that one organization is inherently biased, having attestation from an organization sharing their preferred ideology should help convince them of the leak's authenticity.

7.1.7 Attestation Operation

The attestor is modelled as a computer server that responds to queries with commitments and message sets. If the server automatically responds to these requests, then it could become vulnerable to 'denial of service' attacks. These attacks flood the server with trivial requests in an attempt to take it off-line. A public server could also receive requests for groups and organizations that its owner does not want it to attest for. This could occur when a server attests to a leak from within its own organization. In practice we suggest that the server be managed by a human actor, who decides based on the contents of the initial message from the source whether the protocol should continue or not.

It is also possible that the attestation server posts a list of public credentials that it has attested to, instead of responding to skeptic inquires like we have modelled in sub-ceremony VER. In doing so, the burden of responding to all of the verification requests is removed. The timing of when the server posts these credentials would be important, as it should not do so before the leak has occurred. Otherwise there is additional risk that an adversary is able to use that information to create a false leak. Depending on the scheme used, a public posting could then invalidate all future leaks

under that credential.

7.2 Future Work

A major future goal is to implement the presented constructions, and experimentally evaluate them. First we should consult with potential users to further refine the requirements of a practical secret leaking system before finalizing a construction. Then we can seek additional support to determine exactly what resources are required to build the system. Once these steps have been completed, we can we deploy the implementation for experimental purposes. By carrying out experiments, we will be able to evaluate the robustness of an implementation against additional attacks.

Another future goal is to determine if we can reach a better lower bound for the adversary's ability to re-identify the source. In the constructions we have presented in this work, we were only able to demonstrate that they provide a $\frac{2}{n}$ guarantee versus the theoretical lower bound of $\frac{1}{n}$. The theoretical lower bound would be achievable if the source has the ability to verify that the other group members received the same message. At this time we have not been able to devise a means of doing this without introducing additional vulnerabilities that could lead to forgeries of the credential.

We can also study other potential applications for the general protocol in addition to secret leaking. It would be possible to design and build a reviewing system using this protocol. In the case of course reviews at a university, the group would be the class of students and the challenges would be sent to the student's email. In the case of an employer review website, the group would be the current employees and the challenges would be sent to emails associated with the employer. In both cases, such a system would provide greater privacy to the reviewer than current systems while

allowing an auditor to have some publicly verifiable means of checking the review. There exist other approaches to provable security outside of the reductionist proofs we used in this work. Security protocol verification tools, such as a Tamarin [MSCB13] or ProVerif [KBB17] allow for protocol security to be investigated using formal methods techniques. We can explore using in this approach to futher study and evaluate the security properties of the attested drop protocol.

7.3 Conclusion

In this work, we have presented a new protocol which allows for anonymous attestation without a PKI. This protocol can be used in scenarios where the user wants to maintain its privacy from all parties, while still being able to provide to a skeptic some evidence that the document they have provided is from an authentic source. These scenarios come up frequently when the user is a whistleblower, so this work can be developed into a system to help protect these individuals from serious consequences. We also looked at the potential threats and trust assumptions needed for an attested drop protocol. While the system does not provide proof that the contents of the document used with it are authentic, it does provide a means of validating where the document came from, so long as the skeptic trusts at least one of the attestors to act honestly. An implementation of the protocol was not provided, but with consultation with potential users, future research will hopefully result in the deployment of a real-world system.

While we have created a protocol for anonymous attestation, we have not spent much time addressing the implications to society if an implementation of this protocol was deployed. With stronger whistleblowing systems, it will become easier for honest

individuals to expose corruption and mismanagement in both the public and private sphere. This will hopefully allow for stronger regulation, and through it better government and management. News agencies could use this system to improve the verifiability of their sources without having to reveal their identities. With widespread adoption, this protocol could lead to a more open and honest world.

Bibliography

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *USENIX Security 2008*, pages 335–348. USENIX Association, July / August 2008.

[AFNV19] Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 701–731, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-26951-7_24.

[Ano18] Anonymous. I am part of the resistance inside the Trump administration. *The New York Times*, 2018. URL: https://www.nytimes.com/2018/09/05/opinion/trump-white-house-anonymous-resistance.html.

[Aut19] Philip Authier. Ombudsman says Quebec's whistleblowing laws failed public servant. *Montreal Gazette*, 2019. URL: https://montrealgazette.com/news/quebec/ombudsman-says-quebecs-whistleblowing-laws-failed-public-servant.

[BCS16] Dan Boneh, Henry Corrigan-Gibbs, and Stuart E. Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 220–248. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_8.

[BDK16] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: new generation of memory-hard functions for password hashing and other applications. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 292–302. IEEE, 2016. doi:10.1109/EuroSP.2016.31.

[Bel98] Mihir Bellare. Practice-oriented provable-security (invited lecture). In Eiji Okamoto, George I. Davida, and Masahiro Mambo, editors, *ISW'97*, volume 1396 of *LNCS*, pages 221–231. Springer, Heidelberg, September 1998.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, January 2009. doi:10.1007/s00145-007-9011-9.

[BL18] Sebastian Berndt and Maciej Liskiewicz. On the gold standard for security of universal steganography. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*,

pages 29–60. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78381-9_2.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_25.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, Heidelberg, December 1997.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.

[CL98] Lorrie Faith Cranor and Brian A. LaMacchia. Spam! *Commun. ACM*, 41(8):74–83, August 1998. doi:10.1145/280324.280336.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991. doi:10.1007/3-540-46416-6_22.

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_39.

[Dam07] Ivan Damgård. A "proof-reading" of some issues in cryptography. In Lars Arge, Christian Cachin, Tomasz Jurdziński, and Andrzej Tarlecki, editors, *Automata, Languages and Programming*, pages 2–11, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[DFKP16] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Cinderella: Turning shabby X.509 certificates into elegant anonymous credentials with the magic of verifiable computation. In *2016 IEEE Symposium on Security and Privacy*, pages 235–254. IEEE Computer Society Press, May 2016. doi:10.1109/SP.2016.22.

[DGH12] Casey Devet, Ian Goldberg, and Nadia Heninger. Optimally robust private information retrieval. In Tadayoshi Kohno, editor, *USENIX Security 2012*, pages 269–283. USENIX Association, August 2012.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The

second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[DR] Graham Dawes and Gregory Rammego. Welcome to Deloitte Tip-offs Anonymous. https://www.tip-offs.com/.

[DY91] Alfredo De Santis and Moti Yung. On the design of provably secure cryptographic hash functions. In Ivan Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 412–431. Springer, Heidelberg, May 1991. doi:10.1007/3-540-46877-3_37.

[Ell07] Carl Ellison. Ceremony design and analysis. Cryptology ePrint Archive, Report 2007/399, 2007. http://eprint.iacr.org/2007/399.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984. doi:10.1016/0022-0000(84)90070-9.

[GNF+17] Paul Grassi, Elaine Newton, James Fenton, Ray Perlner, Andrew Regenscheid, William Burr, Justin Richer, Naomi Lefkovitz, Jamie Danker, Yee-Yin Choong, Kristen Greene, and Mary Theofanos. Digital identity guidelines: Authentication and lifecycle management. Technical report, NIST, June 2017. URL: https://doi.org/10.6028/NIST.SP.800-63b.

[HJ07] Jaap-Henk Hoepman and Bart Jacobs. Increased security through open source. *Commun. ACM*, 50(1):79–83, January 2007. doi:10.1145/1188913.1188921.

[KBB17] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2nd IEEE European Symposium on Security and Privacy (EuroS&P'17)*, pages 435–450, Paris, France, April 2017. IEEE.

[KES+16] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg. SoK: Making sense of censorship resistance systems. *PoPETs*, 2016(4):37–61, October 2016. doi:10.1515/popets-2016-0028.

[KKM+12] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *2012 IEEE Symposium on Security and Privacy*, pages 523–537. IEEE Computer Society Press, May 2012. doi:10.1109/SP.2012.38.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007. doi:10.1007/s00145-005-0432-z.

[KM17] A. Rusch K. Moriarty, B. Kaliski. PKCS 5: Password-Based Cryptography Specification Version 2.1. RFC 8018, RFC Editor, January 2017. URL: https://www.rfc-editor.org/rfc/rfc8018.txt.

[Kol19] Sheelah Kolhatkar. The personal toll of whistle-blowing. *The New Yorker*, 2019. URL: https://www.newyorker.com/magazine/2019/02/04/the-personal-toll-of-whistle-blowing.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_34.

[KRRS14] Aggelos Kiayias, Yona Raekow, Alexander Russell, and Narasimha Shashidhar. A one-time stegosystem and applications to efficient covert communication. *Journal of Cryptology*, 27(1):23–44, January 2014. doi:10.1007/s00145-012-9135-4.

[KSJS07] Mark Keith, Benjamin Shao, and Paul John Steinbart. The usability of passphrases for authentication: An empirical field study. *International Journal of Human-Computer Studies*, 65:17–28, 01 2007. doi:10.1016/j.ijhcs.2006.08.005.

[KSK+11] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge

Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 2595–2604. ACM, 05 2011. doi:10.1145/1978942.1979321.

[KZZ17] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Ceremonies for end-to-end verifiable elections. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 305–334. Springer, Heidelberg, March 2017. doi:10.1007/978-3-662-54388-7_11.

[Law09] Nate Lawson. Side-channel attacks on cryptographic software. *IEEE Security & Privacy*, 7(6):65–68, 2009.

[LMP+12] Patrick Lincoln, Ian Mason, Phillip Porras, Vinod Yegneswaran, Zachary Weinberg, Jeroen Massar, William Simpson, Paul Vixie, and Dan Boneh. Bootstrapping communications into an anti-censorship system. In *Presented as part of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, Bellevue, WA, 2012. USENIX. URL: https://www.usenix.org/conference/foci12/workshop-program/presentation/Lincoln.

[LT] Kai Leisering and Kenan Tur. Business Keeper: Your Security for Whistleblower. https://www.business-keeper.com/en/.

[MSCB13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 696–701. Springer Berlin Heidelberg, 2013.

[NB74] R. Nader and Blackwell. *Whistle Blowing*. Penguin Group (USA) Incorporated, 1974. URL: https://books.google.ca/books?id=Onfk_1zF08IC.

[PHE+17] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security 2017*, pages 1199–1216. USENIX Association, August 2017.

[PM99] Niels Provos and David Mazieres. A future-adaptable password scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91, 1999.

[RBGNB11] Kenneth Radke, Colin Boyd, Juan Gonzalez Nieto, and Margot Brereton. Ceremony analysis: Strengths and weaknesses. In *IFIP Advances in Information and Communication Technology*, volume 354, pages 104–115, 01 2011. doi:10.1007/978-3-642-21424-0_9.

[Rey06] Paul Reynolds. Old spying lives on in new ways. *BBC News*, 2006. URL: http://news.bbc.co.uk/2/hi/europe/4639758.stm.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001. doi:10.1007/3-540-45682-1_32.

[Sch14] William E Scheuerman. Whistleblowing as civil disobedience: The case

of Edward Snowden. *Philosophy & Social Criticism*, 40(7):609–628, 2014.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[SKK+12] Richard Shay, Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Blase Ur, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Correct horse battery staple: Exploring the usability of system-assigned passphrases. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 7:1–7:20, New York, NY, USA, 2012. ACM. doi:10.1145/2335356.2335366.

[SPD] Aaron Swartz, Kevin Poulson, and James Dolan. SecureDrop Documentation. https://docs.securedrop.org/en/release-0.9/.

[SW07] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 166–180. Springer, Heidelberg, April 2007. doi:10.1007/978-3-540-71677-8_12.

[TBBC10] Meltem Sonmez Turan, Elaine Barker, William Burr, and Lily Chen. Recommendation for password-based key derivation: Part 1: Storage

applications. Technical report, NIST, December 2010. URL: https://csrc.nist.gov/publications/detail/sp/800-132/final.

[Tor] Tor Project. Tor Metrics. https://metrics.torproject.org/userstats-relay-country.html.

[Vac04] J. Vacca. *Public Key Infrastructure*. Auerbach Publications, New York, 2004. doi:10.1201/9780203498156.

[VDHLZZ15] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[VMBM19] Andrea Visconti, Ondrej Mosnek, Milan Broz, and Vashek Maty. Examining pbkdf2 security margin-case study of LUKS. *Journal of Information Security and Applications*, 46:296 – 306, 2019. doi:10.1016/j.jisa.2019.03.016.

[WAP+18] Liang Wang, Gilad Asharov, Rafael Pass, Thomas Ristenpart, and abhi shelat. Blind Certificate Authorities. Cryptology ePrint Archive, Report 2018/1022, 2018. https://eprint.iacr.org/2018/1022.

[WWY+12] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. StegoTorus: a camouflage proxy for the Tor anonymity system. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 109–120. ACM Press, October 2012. doi:10.1145/2382196.2382211.

[YY05] Frances F. Yao and Yiqun Lisa Yin. Design and analysis of password-based key derivation functions. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 245–261. Springer, Heidelberg, February 2005. doi:10.1007/978-3-540-30574-3_17.