Lattice-based DAPS and Generalizations: Self-Enforcement in Signature Schemes

Dan Boneh, Sam Kim, and Valeria Nikolaenko

Stanford University

Abstract. Double authentication preventing signatures (DAPS) is a mechanism, due to Poettering and Stebila, for protecting certificate authorities (CAs) from coercion. We construct the first lattice-based DAPS signatures, thereby providing the first post-quantum DAPS system. We go further and generalize DAPS to a more general mechanism we call *predicate* authentication preventing signatures (PAPS). Here, for a given k-ary predicate ϕ, a PAPS system for ϕ is regular signature scheme. However, if the signer ever signs k messages $m_1, \ldots, m_k$ such that $\phi(m_1, \ldots, m_k)$ is true then these k signatures reveal the signer's secret key. This self-enforcement mechanism incentivizes the signer to never sign conflicting messages, namely messages that satisfy the predicate ϕ. The k conflicting messages can be signed at different times and the signatures may be generated independently of one another. We further generalize to the case when the signatures are generated by multiple signers. We motivate these primitives, give precise definitions, and provide several constructions. These primitives are challenging to construct and give rise to many new elegant open research questions.

1 Introduction

Suppose a web site such as `facebook.com` buys certificates for its domain from a certificate authority (CA) called XYZ. These certificates enable web browsers to establish a (one-sided) authenticated session with `facebook.com`. Sometime later, a law enforcement agency or a nation state that has jurisdiction over the CA compels XYZ to secretly issue a fresh certificate for `facebook.com`. The CA has no choice but to comply. The agency can then use this issued certificate in a man-in-the-middle attack on `facebook.com`. Web users have no way to detect that this is happening and that their traffic is being intercepted. We emphasize that the rogue certificate is issued by the same CA from which `facebook.com` normally buys its certificates. The only user-side signal is that a previously unseen public-key is being served in a `facebook.com` certificate, but this happens frequently under normal operation at a large site and would not generally look suspicious.

Some technologies, such as certificate transparency (CT) [LLK15] as well as CONIKS [MBB$^+$15], are designed to detect situations where a CA such as XYZ issues a fake certificate for a domain. These technologies empower an origin domain, `facebook.com` in this case, to detect that a fake certificate was issued for its domain.

Poettering and Stebila [PS14] proposed a very different defense against the scenario described above. Their idea, called double-authentication-preventing signatures, or DAPS for short, is as follows: suppose XYZ signs all its certificates using a signature scheme where the signing algorithm uses the secret signing key sk to sign a pair $(\mathsf{subj}, \mathsf{payload})$. Here subj is the domain-name to which the certificate is issued and $\mathsf{payload}$ is all other fields in the certificate. The resulting signature σ can be verified as a standard digital signature. The key property of DAPS is the following: suppose XYZ publishes two valid signatures σ_1 and σ_2 *for the same* subj *but for different payloads*, say one on $(\mathsf{subj}, \mathsf{payload}_1)$ and another on $(\mathsf{subj}, \mathsf{payload}_2)$. Then these two signatures enable anyone to expose XYZ's secret signing key sk. The point is that XYZ can argue that it should not be forced to issue the rogue certificate for `facebook.com` because that would expose its signing key thereby causing massive collateral damage to *all* of XYZ's customers. Whether this argument is effective remains to be seen, but the idea itself is interesting and, as we show below, leads to interesting and challenging cryptographic questions.

How to use DAPS. There are many practical holes in the basic DAPS proposal described above that prevent it from being used as is, but with a bit of thought they can be addressed. However, our goal here is not to argue that DAPS will be deployed in practice, but rather to motivate this as an interesting cryptographic question. Towards this goal, we examine broader applications of DAPS as well as an elegant generalization.

Other applications for DAPS. Beyond certificates, DAPS can be a useful "self-enforcement" security mechanism. For example, suppose Eve owns a certain patent and wants to sell the rights to the patent. Bob wants to buy the patent from Eve, but he is worried that Eve will sell the patent to multiple people. Using DAPS, Eve can use the patent number as the subject and use "owned by Bob" as the payload. If she tries to sell the same patent to two different people she will end up signing two pairs of messages with the same subject, but different payload. The resulting two signatures can be combined to expose Eve's private key. If this private key is of high value to Eve then this self-enforcement mechanism will prevent her from double-selling the same patent to two people. This way, Bob has some confidence in the exclusivity of the deal with Eve.

PAPS: DAPS for general predicates. The previous paragraph motivates a more general elegant primitive which we call *predicate-authentication-preventing signatures*, or PAPS for short. Let $\mathcal{M}$ be a message space and let $\phi : \mathcal{M}^k \to \{0,1\}$ be a predicate. A PAPS scheme for ϕ lets the signer sign any message $m \in \mathcal{M}$, just as in a regular signature scheme. However, if over the life of the secret key, the signer signs messages $m_1, \ldots, m_k \in \mathcal{M}$ such that $\phi(m_1, \ldots, m_k) = 1$ then these k signatures can be combined to expose the signer's secret signing key. This secret key extraction should work no matter how the k signatures are generated: as long as all k signatures are valid, it should be possible to extract the secret key.

For security, as long as the predicate ϕ is never satisfied, the signature scheme should be existentially unforgeable under a chosen message attack, as for regular signatures.

Notice that DAPS is a special case of PAPS: the key space $\mathcal{M}$ is a set of pairs $\mathcal{M} = \mathcal{S} \times \mathcal{P}$ and the predicate ϕ_{DAPS} is simply the 2-ary predicate:

$$\phi_{\text{DAPS}}\big(\ (x_1, y_1),\ (x_2, y_2)\ \big) = 1 \quad \Leftrightarrow \quad x_1 = x_2 \text{ and } y_1 \neq y_2.$$

More general predicates come up naturally. For example, suppose a web site owns k machines and it wants to generate a different key-pair for each machine, necessitating a different certificate for each machine. The analogue of DAPS is a k-way DAPS where the message space $\mathcal{M}$ is again $\mathcal{M} = \mathcal{S} \times \mathcal{P}$, but now the predicate ϕ is the $(k+1)$-ary predicate

$$\phi\big(\ (x_1, y_1), \ldots, (x_{k+1}, y_{k+1})\ \big) = 1 \quad \Leftrightarrow \quad \begin{Bmatrix} x_1 = \cdots = x_{k+1} \text{ and} \\ y_1, \ldots, y_{k+1} \text{ are all distinct.} \end{Bmatrix}$$

This lets the site use a different certificate for each of its k machines, but if another certificate is issued then the CA's secret key is exposed. We give a construction for this predicate in Section 5 as well as for several other predicates.

Proving security of a PAPS construction is non trivial. For example, suppose the message space is $\mathcal{M} = \mathbb{F}_p$ for some prime p. Consider the 3-ary predicate ϕ defined as $\phi(x, y, z) = 1$ if and only if $x + y + z = 0$. That is, the secret key should leak if the signer ever signs three messages whose sum is zero. However, if the signer never signs three messages satisfying this condition, the signature scheme should be existentially unforgeable. To prove existential unforgeability, the simulator must interact with the adversary, answering all the adversary's adaptive signature queries, and using the adversary's existential forgery to solve a challenge problem. The problem is that, because the adversary's first two queries can be for arbitrary messages, the simulator must be prepared to provide a signature for *all* messages $m \in \mathcal{M}$. In particular, the simulator will know the signature on three messages x, y, z satisfying $x+y+z = 0$. But then the simulator can extract the secret signing key, and can produce any forgery by itself, meaning that the adversary is not helping the simulator. Nevertheless, in Section 5 we are able to prove security for several generalized predicates, though not for the 3-way summation predicate.

A further generalization: multi-signer PAPS. We can further generalize the notion of PAPS to the setting of k signers where each signer has its own signing/public key-pair. As before, let ϕ be a k-ary predicate $\phi : \mathcal{M}^k \to \{0, 1\}$. Suppose that for $i = 1, \ldots, k$ signer number i signs message $m_i \in \mathcal{M}$. Then, if $\phi(m_1, \ldots, m_j) = 1$, then these k signatures (along with the k messages and k public keys) can be used to expose some secret s chosen at setup time.

Multi-signer PAPS come up naturally when considering certificates. Suppose that the agency, instead of asking XYZ to issue the rogue certificate for `facebook.com`, it asks a different CA to provide a certificate for `facebook.com`. We are now in a 2-signer scenario. If the predicate ϕ_{DAPS} can be made to work on the two signatures, despite them being from different CAs, then going to a different CA will not help the agency. We define multi-signer PAPS in Section 6 where we also give several constructions.

1.1 Contributions

In this work we build a lattice-based DAPS construction based on Short Integer Solutions (SIS) problem and the Learning with Errors (LWE) problem. Our construction builds upon the structure of the fully homomorphic signature scheme of Gorbunov et al.[GVW15]. In their construction, a signature consists of a preimage of a specially formed target matrix of a lattice trapdoor function. To make key leakage a feature rather than a form of insecurity, we carefully hash the messages to derive the target matrix such that two different message of the same subject leads to two matrices for which two preimage matrices leak a trapdoor. As in [PS14], we prove security in the random oracle model.

Also, as we discuss above, we extend DAPS to a more general primitive that we call predicate authentication preventing signatures (PAPS). In this setting, signatures of any messages that satisfy a certain predicate defined on these messages leak a signer's secret key. To motivate the notion, we show that for certain simple, but useful predicates, PAPS can already be constructed from DAPS.

Finally, we further extend PAPS to a multi-authority settings where signatures from different signers can also leak some shared private information. We give formal definitions in this setting and show that our lattice DAPS construction can be extended to this setting as well using the property that two short preimages of a specifically formed target matrix of lattice trapdoor functions can be merged to give a trapdoor of an extended lattice.

1.2 Related Work

Previous works on DAPS. The notion of double-authentication-preventing signatures was introduced by Poettering and Stebila [PS14]. They provide a construction based on extractable trapdoor functions that can be constructed using the group of quadratic residues modulo a Blum integer. Subsequently, Bellare, Poettering, and Stebila [BPS17] gave a generic construction based on trapdoor identification schemes where the private randomness committed by the prover can be extracted using a trapdoor. We note that although lattice-based identification schemes have appeared in the literature [Lyu08,Lyu12], the construction from [BPS17] does not directly give a lattice-based DAPS construction since lattice trapdoors are randomized with multiple preimages. Constructing DAPS from lattice-based assumptions is an interesting and important goal since they provide hardness even against quantum computers, a setting for which the previous two works do not provide security.

Delegating restricted signing keys. A number of works in the literature have focused on schemes that allow an authority to delegate signing keys with some restricted functionalities (with function privacy). These include attribute-based signatures [MPR11], functional signatures [BGI14] and policy-based signature [BF14]. In these schemes, a signer is restricted to sign only certain messages that satisfy a predicate requirement. One difference between these notions and DAPS/PAPS is that in the former, the restriction is done by a central authority to restrict other signers, while in the latter, the authority restricts itself as a self-enforcement mechanism. Another major difference is that in the former, the restriction is determined with respect to each individual message while in the latter, the restriction is determined by all of the past messages that the signer signs, which is what makes DAPS and PAPS an interesting theoretical notion.

Ring/Group signatures. A similar notion of double-signing preventing mechanism exists in the setting of group signatures [TX03] and ring signatures [RST01,BKM06] called Revocable-iff-Linked (RiffL) signatures [ALSY06]. In this setting, a signer can sign on behalf of a group; however, if it signs twice or more, then the identity of the signer is leaked. As in the discussion of the previous paragraph, one difference in the DAPS setting compared to RiffL is that DAPS is a self-enforcing mechanism, which means that the linkability is not enforced by another trusted authority of the system, but by itself. However, the more fundamental difference is that in DAPS, the act of double-signing immediately gives away the signing key or private data rather than simply leaking the information that it double signed. As was discussed in [PS14], there are instances where simply leaking the fact that a CA double signed may not be enough of a penalty (i.e. the 2011 Comodo incident[1]) and DAPS is designed to cope with these type of situations.

2 Preliminaries

Basic Notation. For an integer N, we write $[N]$ to denote the set $\{1, ..., N\}$. We use bold lowercase letters (e.g., $\mathbf{x}, \mathbf{w}$) to denote vectors and bold uppercase letters (e.g., $\mathbf{A}, \mathbf{G}$) to denote matrices. For a matrix $\mathbf{A}$, we use $\mathbf{A}^T$ to denote the trasnpose of $\mathbf{A}$ and for a vector $\mathbf{x}$, we use $\|\mathbf{x}\|$ to denote its Euclidean norm. In general, we write λ for the security parameters. We say a function $\epsilon(\lambda)$ is negligible in λ, if $\epsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$, and we write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We say that an event occurs with *negligible probability* if the probabilty of the event is $\mathsf{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability.

Entropy and Statistical Distance. The *statistical distance* between two random variables X and Y over a finite domain Ω is defined as

$$\mathsf{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} \left| \Pr[X = \omega] - \Pr[Y = \omega] \right| .$$

[1] `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`

We say that two distribution ensembles $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are *statistically indistinguishable*, denoted $\overset{\text{stat}}{\approx}$, if it holds that $\mathsf{SD}(X_\lambda, Y_\lambda)$ is negligible in λ. The *min-entropy* of a random variable X, denoted $\mathbf{H}_\infty(X|Y)$, is defined as

$$\mathbf{H}_\infty(X|Y) \overset{\text{def}}{=} -\log\left(\underset{y\leftarrow Y}{\mathbf{E}}\left[\max_x \Pr[X = x|Y = y]\right]\right)$$

The optimal probability of an unbounded adversary guessing X given the correlated value Y is $2^{-\mathbf{H}_\infty(X|Y)}$.

2.1 Circular Security

In this section we briefly recall the notion of circular security. A public key encryption (PKE) consists of three algorithms $\Pi_{\mathsf{pke}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ where $\mathsf{PKE.KeyGen}$ takes in a unary representation of the security parameter λ and outputs a public and secret key pair $(\mathsf{pk}, \mathsf{sk})$. The encryption algorithm takes in a public key pk and a message m and generates a ciphertext ct. The decryption algorithm takes in a secret key sk and a ciphertext ct and outputs a message m. For correctness, we require that for all $\lambda \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, we have that $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{PKE.KeyGen}(\mathsf{pk}, m)) = m$ with overwhelming probability.

Definition 1 (Circular Security [CL01,BRS02]). *A public-key encryption scheme Π_{pke} is* circular secure *if for all efficient adversaries $\mathcal{A}$, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{circ}}_{\Pi_{\mathsf{pke}},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left|\Pr[\mathsf{Expt}^{(0)}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where for each $b \in \{0, 1\}$, and $\lambda \in \mathbb{N}$, the experient $\mathsf{Expt}^{(b)}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
2. $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, 0^{|\mathsf{sk}|})$.
3. $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, \mathsf{sk})$.
4. $b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}_b)$.
5. *Output* $b' \in \{0, 1\}$.

Circular security assumption on lattice-based cryptosystems have been used extensively throughout the literature [Gen09,BV11,BV14,BGV12,GSW13] with some positive results showing that some common forms of lattice-based encryption schemes can be shown to be circular secure [ACPS09,ASP12].

2.2 Broadcast Encryption

A broadcast encryption scheme BE [FN93] consists of a tuple of algorithms $\Pi_{\mathsf{be}} = (\mathsf{BE.KeyGen}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ defined as follows:

1. $\mathsf{BE.KeyGen}(1^\lambda, N) \to (\{\mathsf{sk}_i\}_{i\in[N]}, \mathsf{pk})$: On input the security parameter λ and a positive integer $N \in \mathbb{N}$, the key generation algorithm outputs a set of secret keys $\{\mathsf{sk}_i\}_{i\in[N]}$ and a public key pk.
2. $\mathsf{BE.Encrypt}(\mathsf{pk}, \mathsf{msg}, T) \to \mathsf{ct}_T$: On input a public key pk, a message m, and a set of intended recipients $T \subseteq [N]$, the encryption algorithm outputs a ciphertext ct_T.
3. $\mathsf{BE.Decrypt}(\mathsf{sk}_i, \mathsf{ct}) \to m'$: On input a secret key sk_i and a ciphertext ct, the decryption algorithm outputs a message m'.

Correctness. For correctness we require that for all $\lambda \in \mathbb{N}$, $N \in \mathbb{N}$, $T \subseteq [N]$, $(\{\mathsf{sk}_i\}_{i\in[N]}, \mathsf{pk}) \leftarrow \mathsf{BE.KeyGen}(1^\lambda, N)$, we have $\mathsf{BE.Decrypt}(\mathsf{sk}_i, \mathsf{BE.Encrypt}(\mathsf{pk}, \mathsf{ct}, T)) = m$ for all $i \in T$.

Security. For broadcast encryption scheme, we define the following security notion.

Definition 2. *A broadcast encryption scheme Π_{be} is secure if for all efficient adversaries $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\lambda)$ such that*

$$\mathsf{Adv}_{\Pi_{\mathsf{be}},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left|\Pr[\mathsf{Expt}^{(0)}_{\mathsf{BE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\mathsf{BE},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where for each $b \in \{0,1\}$, and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Expt}^{(b)}_{\mathsf{BE},\mathcal{A}}$ is defined as follows:

– $(\{\mathsf{sk}_i\}_{i\in[N]}, \mathsf{pk}) \leftarrow \mathsf{BE.KeyGen}(1^\lambda, N)$.
– $(T, m_0, m_1) \leftarrow \mathcal{A}_0(\mathsf{pk})$.
– $\mathsf{ct}_b \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, m_b)$.
– $b' \leftarrow \mathcal{A}_1(\{\mathsf{sk}_i\}_{i\in[N]\setminus T}, \mathsf{ct}_b)$.
– *Output* $b' \in \{0,1\}$.

A number of constructions for broadcast encryption schemes have been proposed in the literature [FN93,BGW05,BWZ14] with short ciphertext where the length of the ciphertext scales sublinearly in the number of users in the system. For linear length ciphertext, a broadcast encryption scheme can be constructed generically from a regular public key encryption scheme by concatenating the N instances of the public key encryption schemes.

2.3 Background on Lattices

In this section, we describe some of the results and notations for lattice-based cryptography that are used throughout the paper.

Lattice and Gaussians Let n, q, m be positive integers. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^{\perp}(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \mod q\}$. More generally, for $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the shifted lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \mod q\}$.

Regev [Reg09] defined a natural distribution on $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ called a *discrete Gaussian* parameterized by a scalar $s > 0$. We use $\mathcal{D}_s(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ to denote this distribution. For a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $s > \widetilde{O}(\sqrt{n})$, a vector $\mathbf{x}$ sampled from $\mathcal{D}_s(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ has Euclidean norm less than $s\sqrt{m}$ with overwhelming probability. For a matrix $\mathbf{U} = (\mathbf{u}_1 | \dots | \mathbf{u}_k) \in \mathbb{Z}_q^{n \times k}$, we let $\mathcal{D}_s(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ be a distribution on matrices in $\mathbb{Z}^{m \times k}$ where the i-th column is sampled from $\mathcal{D}_s(\Lambda_q^{\mathbf{u}_i}(\mathbf{A}))$ for $i = 1, ..., k$.

The SIS Problem. Let n, m, q, β be positive integers. In the SIS(n, m, q, β) problem, the adversary is given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its goal is to find a vector $\mathbf{u} \in \mathbb{Z}_q^m$ with $\mathbf{u} \neq \mathbf{0}$ and $\|\mathbf{u}\| \leq \beta$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$.

The SIS problem is known to be as hard as certain worst-case lattice problems. In particular, for any $m = \mathsf{poly}(n)$, any $\beta > 0$, and any sufficiently large $q \geq \beta \cdot \mathsf{poly}(n)$, solving SIS$(n, m, q, \beta)$ is at least as hard as approximating certain worst-case lattice problems such as the Shortest Vector Problem (GapSVP) and the Short Independent Vectors Problem (SIVP) on n-dimensional lattices to within $\beta \cdot \mathsf{poly}(n)$ factor [Ajt96,Mic04,MR07,MP13]. The hardness of SIS is also implied by the LWE problem.

The LWE Problem. Let n, m, q be positive integers and χ a noise distribution over $\mathbb{Z}_q$. In the LWE(n, m, q, χ) problem, the adversary's goal is to distinguish between the two distributions:

$$(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e}) \qquad \text{and} \qquad (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^m$ are uniformly sampled.

We say that a noise distribution χ is B-bounded if its support is in $[-B, B]$. For any fixed $d > 0$, and sufficiently large q, taking χ as a certain q/n^d-bounded distribution, the LWE(n, m, q, χ) problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices to within $\mathsf{poly}(n)$ factor [Reg09,Pei09,ACPS09,MM11,MP12,BLP$^+$13].

Matrix Norms. For a matrix $\mathbf{R} \in \mathbb{Z}^{k \times m}$, we define the matrix norms:

- $\|\mathbf{R}\|$ denotes the ℓ_2 length of the longest column of $\mathbf{R}$.
- $\|\mathbf{R}\|_2$ is the operator norm of $\mathbf{R}$ defined as $\|\mathbf{R}\|_2 = \sup_{\|\mathbf{x}\|=1} \|\mathbf{R}\mathbf{x}\|$.

Note that always $\|\mathbf{R}\| \leq \|\mathbf{R}\|_2 \leq \sqrt{k}\|\mathbf{R}\|$ and that $\|\mathbf{R} \cdot \mathbf{S}\|_2 \leq \|\mathbf{R}\|_2 \cdot \|\mathbf{S}\|_2$.

Lattice Trapdoors. Here, we review the known results about lattice trapdoors which make the SIS and LWE problems easy to solve with knowledge such trapdoor. For this work, it is convenient to work with the notion of "gadget" based trapdoors as formalized in [MP12]. In such setting, there is a structured public gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times n\ell}$ for $\ell = \lceil \log q \rceil$. A trapdoor for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is an integer matrix $\mathbf{R} \in \mathbb{Z}_q^{n\times n\ell}$ such that $\mathbf{AR} = \mathbf{HG}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$. The quality of a trapdoor is measured by the operator norm of $\mathbf{R}$ where the smaller norm $\|\mathbf{R}\|_2$ means higher quality. We will often use the symbol $\mathbf{T}_\mathbf{A}$ to denote the trapdoor matrix $\mathbf{R}$.

Since the exact constructions and algorithm details of such trapdoors are not needed for this work, we abstract out these details and summarize the relevant results in the lemma below.

Lemma 1 ([Ajt96,GPV08,AP11,MP12]). *There exist polynomial time algorithms* $\mathsf{TrapGen}, \mathsf{SamPre}, \mathsf{Sam}, \mathsf{Invert}$ *such that the following holds. Given positive integers $n \geq 1, q \geq 2$, there exists $m^* = O(n \log q)$ such that for $k = \mathsf{poly}(n)$, we have:*

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, \mathbf{T}_\mathbf{A})$*: A randomized algorithm that when $m \geq m^*$, outputs a full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a trapdoor $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m\times n\ell}$ such that $\mathbf{A}$ is statistically close to uniform and $\|\mathbf{R}\|_2 = O(\sqrt{m})$.*
- $\mathsf{SamPre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{V}, s) \to \mathbf{U}$*: A randomized algorithm that on input $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a trapdoor $\mathbf{T}_\mathbf{A}$ of $\mathbf{A}$, a matrix $\mathbf{V} \in \mathbb{Z}_q^{m\times k}$, and $s = \widetilde{O}(\|\mathbf{T}_\mathbf{A}\|_2)$, outputs a random sample $\mathbf{U} \in \mathbb{Z}^{m\times k}$ from the distribution $\mathcal{D}_s(\Lambda_q^\mathbf{V}(\mathbf{A}))$.*
- $\mathsf{Sam}(1^m, 1^k, q, s) \to \mathbf{U}$*: A randomized algorithm that samples a matrix $\mathbf{U} \in \mathbb{Z}^{m\times k}$ such that each of its column is sampled from $\mathcal{D}_{\mathbb{Z}^m,s}$. We have that for $s \geq \omega(\sqrt{\log m})$ the matrix $\mathbf{V} = \mathbf{A}\cdot\mathbf{U}$ is statistically close to a uniform matrix in $\mathbb{Z}_q^{n\times k}$ and furthermore, the distribution of $\mathbf{U}$ given $\mathbf{V}$ is $\mathcal{D}_s(\Lambda_q^\mathbf{V}(\mathbf{A}))$.*
- $\mathsf{Invert}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{b}) \to \mathbf{s}$*: A deterministic algorithm that on input $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a trapdoor $\mathbf{T}_\mathbf{A}$ of $\mathbf{A}$, and an LWE vector $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$ for $\|\mathbf{e}\| \leq q/O(\|\mathbf{T}_\mathbf{A}\|_2)$, outputs the unique secret vector $\mathbf{s}$.*

To simplify the notation, thoughout the paper, we will always assume that the gadget matrix $\mathbf{G}$ has the same width m as the matrix $\mathbf{A}$ output by the algorithm $\mathsf{TrapGen}$.

2.4 FRD Encoding

In this section, we review an encoding function $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ that maps vectors in $\mathbb{Z}_q^n$ to invertible matrices in $\mathbb{Z}_q^{n\times n}$ with the property that for any two distinct vectors $\mathbf{u}$ and $\mathbf{v}$, the difference between the outputs $H(\mathbf{u})$ and $H(\mathbf{v})$ is never singular, i.e., $\det(H(\mathbf{u}) - H(\mathbf{v})) \neq 0$.

Definition 3. *Let q be a prime and n a positive integer. We say that an efficiently computable function $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ is a* full-rank difference *(FRD) encoding scheme if for all distinct $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$, the matrix $H(\mathbf{u}) - H(\mathbf{v}) \in \mathbb{Z}_q^{n\times n}$ is full rank.*

This notion was formalized in [ABB10] and an injective encoding function satisfying the definition was explicitly constructed by generating an additive subgroup $\mathcal{G}_{\mathrm{FRD}}$ of full-rank matrices by embedding ring multiplications into matrices (similar techniques were used in [CD09,PR06,LM06]). We do not explicitly provide the construction here and mainly use the result as a black box throughout this work.

3 Predicate-authentication-preventing Signatures

In this section, we formally define the notion of *predicate authentication preventing signatures* (PAPS) which generalizes the notion of *double authentication preventing signatures* (DAPS) that was introduced in [PS14].

3.1 PAPS Framework

The syntax for predicate-authentication-preventing signatures largely coincides with the syntax for standard signature schemes with an additional extraction algorithm that can extract some private information of the signer (i.e. the signing key) given the signatures of messages that satisfy a particular predicate.

Definition 4 (PAPS). *A* predicate-authentication-preventing signature *(PAPS) on a corresponding message space $\mathcal{M}$, and a predicate $f : \mathcal{M}^k \to \{0,1\}$ is a tuple of efficient algorithms $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.Extract})$ defined as follows:*

- $\mathsf{PAPS.KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$*: On input a security parameter 1^λ, the key generation algorithm* $\mathsf{PAPS.KeyGen}$ *outputs a signing key* sk *and a verification key* vk*.*
- $\mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg}) \to \sigma$*: On input a signing key* sk *and a message* $\mathsf{msg} \in \mathcal{M}$*, the signing algorithm* $\mathsf{PAPS.Sign}$ *outputs a signature σ.*
- $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}, \sigma)$*: On input a verification key* vk*, a message* $\mathsf{msg} \in \mathcal{M}$*, and a signature σ, the verification algorithm* $\mathsf{PAPS.Verify}$ *accepts/rejects.*
- $\mathsf{PAPS.Extract}(\mathsf{vk}, \{(\mathsf{msg}_i, \sigma_i)\}_{i\in[k]}) \to \mathsf{sk}'$*: On input a verification key* vk*, and a set of message and signature pairs, the extraction algorithm* $\mathsf{PAPS.Extract}$ *outputs the secret key* sk'*.*

Correctness. We say that a PAPS scheme is correct if, for all $\lambda \in \mathbb{N}$, $\mathsf{msg} \in \mathcal{M}$, and $\sigma \leftarrow \mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg})$, we have that $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$.

3.2 Extraction

As in the case of [PS14], we can consider two notions of key extractability depending on whether the signer generates its keys at setup honestly or adversarially. We call the scenario for which the keys are always generated honestly as the *trusted setup model* and the scenario for which the keys can potentially be generated adversarially as the *untrusted setup model*. Before defining these two notions formally, we first define the notion of a *compromising set of signatures*.

Definition 5 (Compromising set of signatures). *Let $f : \mathcal{M}^k \to \{0,1\}$ be a predicate defined on k messages. Then, for a fixed verification key* vk*, a set of k message/signature pairs $\{(\mathsf{msg}_i, \sigma_i)\}_{i\in[k]}$ is* f-compromising *if each signature σ_i is a valid signature of* msg_i *and the k messages satisfy the predicate f; that is, if* $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}_i, \sigma_i) = 1$ *for all $i = 1, ..., k$ and $f(\mathsf{msg}_1, ..., \mathsf{msg}_k) = 1$.*

We now define formally the two notions of key extractability.

Definition 6 (Extractability in trusted setup). *Fix a predicate $f : \mathcal{M}^k \to \{0,1\}$. We say that a* PAPS *scheme on a message space $\mathcal{M}$ is f-*extractable *in the* trusted setup model *if for all $\lambda \in \mathbb{N}$,* $\mathsf{secret} \in \mathcal{S}$*, for all key pairs* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{PAPS.KeyGen}(1^\lambda)$*, and for all compromising set of signatures* $S = \{(\mathsf{msg}_i, \sigma_i)\}_{i\in[k]}$*, we have that* $\mathsf{PAPS.Extract}(\mathsf{vk}, S) = \mathsf{sk}$ *with overwhelming probability.*

For the untrusted setup model, instead of running the honest key generation algorithm, we allow an adversary to generate the keys along with a set of compromising set of signatures and require that the extraction algorithm succeeds on recovering the signing key from these set of signatures. Formally, we define extractability in the untrusted setup as follows.

Definition 7 (Extractability in untrusted setup). *Fix a predicate $f : \mathcal{M}^k \to \{0,1\}$. We say that a* PAPS *scheme on a message space $\mathcal{M}$ is f-*extractable *if for all efficient adversary $\mathcal{A}$, we have that*

$$\Pr\left(\begin{array}{c}(\mathsf{vk}, S = \{(\mathsf{msg}_i, \sigma_i)\}) \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{sk}' \leftarrow \mathsf{PAPS.Extract}(\mathsf{vk}, S)\end{array} : \begin{array}{c} S\ f\text{-}\textit{compromising} \\ \wedge\ \mathsf{sk}' = \mathsf{sk}\end{array}\right) = 1 - \mathsf{negl}(\lambda)$$

Double-authentication-preventing signatures. The notion of double-authentication-preventing signatures is a special case of PAPS. Specifically, in DAPS, the data to be signed $\mathcal{M}_{\mathsf{DAPS}}$ is split into two parts: a *subject* and a *payload*. Then, we consider the following 2-ary predicate in this message space

$$F_{\mathsf{DAPS}}((\mathsf{subj}_0, \mathsf{payload}_0), (\mathsf{subj}_1, \mathsf{payload}_1)) = \begin{cases} 1 \text{ if } \mathsf{subj}_0 = \mathsf{subj}_1, \mathsf{payload}_0 \neq \mathsf{payload}_1 \\ 0 \text{ otherwise.}\end{cases}$$

The predicate is designed specifically for the certificate authority setting where a CA that signs two different messages pertaining to the same subject is penalized by leaking the CA's signing key. In this work, we will mainly focus on constructing PAPS for this particular predicate function.

Remark. An alternative formulation of extractability is allowing some private information of the signer to be extracted instead of the signing key. In this case, the key generation algorithm can take in some secret information to be leaked by a compromising set of signatures and generate the keys accordingly. This formulation generalizes the notion above where extraction always leaks the signing key and can be more befitting for certain applications (see section 7).

3.3 Unforgeability

The security game for the unforgeability notion for PAPS is similar to the standard unforgeability notion for digital signatures where the adversary has access to a signing oracle and wins if it forges a new signature. However, since PAPS is designed precisely to leak the signing key on a compromising set of signatures, we require that the adversary's queries to the signing oracle is limited to non-compromising sets of signatures.

Definition 8 (Unforgeability). *An f-extractable* PAPS *scheme* $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.Extract})$ *is unforgeable if for all efficient adversary* $\mathcal{A}$*, we have that*

$$\mathsf{Adv}^{\mathsf{uf}}_{\Pi_{\mathsf{paps}},\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{uf}} = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{uf}}$ *is defined as follows:*

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{PAPS.KeyGen}(1^\lambda)$.
2. $(\mathsf{msg}^*, \sigma^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{vk})$.
3. $\mathcal{A}$ *wins if* $\mathsf{Verify}^*(\mathsf{msg}^*, \sigma^*)$ *accepts.*

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *and* $\mathsf{Verify}^*(\cdot,\cdot)$ *are defined as follows:*

– *Oracle* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *maintains a list* SignedList *of all the previous valid queries made by* $\mathcal{A}$*. For a query* msg*, that the adversary makes,* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *checks whether there exists a compromising set of signatures in* $\text{SignedList} \cup \{\mathsf{msg}\}$*. If this is the case, then* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *outputs* $\perp$*. Otherwise, it outputs* $\mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg})$.
– *Verifier* $\mathsf{Verify}^*(\mathsf{msg}, \sigma)$ *accepts if* $\mathsf{msg} \notin \text{SignedList}$ *and* $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}, \sigma)$ *accepts.*

4 DAPS from Lattices

In this section, we describe our DAPS construction. For clarity of exposition, we first describe a variant of the dual-Regev encryption scheme [GPV08] which we will use in our construction as a blackbox. This way, we can abstract out the details of the encryption component and present our DAPS construction in a simpler and more intuitive way. After presenting our DAPS construction, we prove its extractable properties and also show that its security can be based on the security of the encryption scheme and the SIS hardness assumption.

4.1 Trapdoor dual-Regev Encryption

In this section, we describe a simple variant of the dual-Regev encryption scheme [GPV08] that we will use as a blackbox for our DAPS construction. We present the encryption scheme here mainly for clarity of exposition and the only difference between the encryption scheme presented here and the original dual-Regev encryption scheme is the use of full trapdoors as the secret key of the scheme as opposed to a short preimage vector as is the case in [GPV08].

We use n as the security parameter λ. Let m, q be the trapdoor parameters dependent on n (Lemma 1). Let χ be a B-bounded noise distribution where $B = O(q/m)$. We construct a public key encryption scheme $\Pi_{\mathsf{pke}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ as follows:

– $\mathsf{PKE.KeyGen}(1^n)$: On input the security parameters 1^n, the key generation algorithm generates trapdoor $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. It sets the public key $\mathsf{pk} = \mathbf{A}$ and $\mathsf{sk} = \mathbf{T_A}$.

– $\mathsf{PKE.Encrypt}(\mathsf{pk}, m)$: On input the public key pk and a message $m \in \{0,1\}$, the encryption algorithm samples uniformly random vectors $\mathbf{d}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, and an error vector from the noise distribution $\mathbf{e} \leftarrow \chi^{m+1}$. It computes the vector

$$\mathbf{b} = [\mathbf{A} \mid \mathbf{d}]^T \mathbf{s} + \mathbf{e} + [\mathbf{0} \mid \lceil q/2 \rceil \cdot m].$$

It outputs the ciphertext $\mathsf{ct} = (\mathbf{d}, \mathbf{b}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{m+1}$.

– $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})$: On input the secret key sk and a ciphertext $\mathsf{ct} = (\mathbf{d}, \mathbf{b})$, the decryption algorithm parses $\mathbf{b} = (\mathbf{b}_0, b_1)$. It then computes $\mathbf{s} \leftarrow \mathsf{Invert}(\mathbf{A}, \mathbf{T_A}, \mathbf{b})$ and $m' = b_1 - \mathbf{d}^T\mathbf{s}$. It outputs $m \in \{0,1\}$ such that m' is close to $\lceil q/2 \rceil \cdot m$.

Correctness. Let $\mathbf{b} = (\mathbf{b}_0 = \mathbf{A}^T\mathbf{s} + \mathbf{e}_0, b_1 = \mathbf{d}^T\mathbf{s} + e_1)$. The $\mathsf{TrapGen}$ algorithm outputs a trapdoor $\mathbf{T_A}$ such that $\|\mathbf{T_A}\|_2 = O(\sqrt{m})$. This means that for $B = O(q/\|\mathbf{T_A}\|\sqrt{m}) = O(q/m)$, the algorithm $\mathsf{Invert}(\mathbf{A}, \mathbf{T_A}, \mathbf{b})$ for $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$ correctly outputs the secret vector $\mathbf{s}$. Then, $b_1 - \mathbf{d}^T\mathbf{s} = (\mathbf{d}^T\mathbf{s} + e_1 + \lceil q/2 \rceil \cdot m) - \mathbf{d}^T\mathbf{s} = \lceil q/2 \rceil \cdot m + \cdot e_1$ which is correctly decoded for given bound on the error distribution χ.

Remark. We note that the correctness still holds with a different trapdoor $\mathbf{T_A}'$ for which $\mathbf{T_A} \neq \mathbf{T_A}'$ as long as $\mathbf{T_A}'$ is of sufficient quality. In fact, we can flexibly adjust the parameter B for the noise distribution χ of the scheme to allow for correct decryption by a slightly lower quality trapdoor. For instance, if we take the parameter to be $B = O(q/m^{3/2})$, then a trapdoor $\mathbf{T_A}'$ of quality $\|\mathbf{T_A}'\|_2 \leq O(m)$ can correctly decrypt the ciphertext. This property will be used for the extractability of our DAPS construction.

Security. The security reduction easily follows from the security proof of the original dual-Regev encryption scheme and we do not reproduce it here.

Theorem 1. *The PKE scheme above is IND-CPA secure assuming that the $LWE(n, m, q, \chi)$ problem is hard.*

For our DAPS construction, we will also assume that the dual-Regev PKE scheme above is circular secure.

4.2 DAPS Construction

We present our construction for DAPS from lattices. Fix a security parameter $n \in \mathbb{N}$ and let m, q, s be the corresponding trapdoor parameters. We use a hash function $\mathsf{H}_{\mathsf{msg}} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_q^{n \times n}$ that maps arbitrary messages to a full rank matrix in $\mathcal{G}_{\mathrm{FRD}}$ and a hash function $\mathsf{H}_{\mathsf{subj}} : \{0,1\}^* \to \mathbb{Z}_q^{n \times m}$ that maps a subject to an SIS matrix for the scheme. We also use the dual-Regev public key encryption scheme $\mathit{\Pi}_{\mathsf{pke}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ as decribed in Section 4.1 with a B-bounded noise distribution where $B = O(q/m^{3/2})$. We construct $\mathit{\Pi}_{\mathsf{daps}} = (\mathsf{DAPS.KeyGen}, \mathsf{DAPS.Sign}, \mathsf{DAPS.Verify}, \mathsf{DAPS.Extract})$ as follows:

- $\mathsf{DAPS.KeyGen}(1^n)$: On the security parameter 1^n, the key generation algorithm first runs $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{PKE.KeyGen}(1^n)$. It then encrypts $\mathsf{ct} \leftarrow \mathsf{PKE.Encrypt}(\mathbf{A}, \mathbf{T_A})$ and set $\mathsf{sk} = \mathbf{T_A}$ and $\mathsf{vk} = (\mathbf{A}, \mathsf{ct})$.
- $\mathsf{DAPS.Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{payload})$: On input the signing key sk, a subject subj, and a message $\mathsf{payload}$, the signing algorithm hashes the message $\mathbf{H}_{\mathsf{msg}} \leftarrow \mathsf{H}_{\mathsf{msg}}(\mathsf{subj}, \mathsf{payload})$ and also hashes the subject $\mathbf{B}_{\mathsf{subj}} \leftarrow \mathsf{H}_{\mathsf{subj}}(\mathsf{subj})$. Then, it computes
$$\mathbf{U} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}_{\mathsf{subj}} + \mathbf{H}_{\mathsf{msg}} \cdot \mathbf{G}, s)$$
where $\mathbf{G}$ is the publicly known gadget matrix. Finally, it outputs $\mathbf{U}$ as the signature.
- $\mathsf{DAPS.Verify}(\mathsf{vk}, \mathsf{subj}, \mathsf{payload}, \sigma)$: On input the verification key vk, a subject subj, a message $\mathsf{payload}$, and a signature $\sigma = \mathbf{U}$, the algorithm hashes the message $\mathbf{H}_{\mathsf{msg}} \leftarrow \mathsf{H}_{\mathsf{msg}}(\mathsf{subj}, \mathsf{payload})$ and also derives the matrix $\mathbf{B}_{\mathsf{subj}} \leftarrow \mathsf{H}_{\mathsf{subj}}(\mathsf{subj})$. Then, the algorithm verifies that
$$\mathbf{A} \cdot \mathbf{U} = \mathbf{B}_{\mathsf{subj}} + \mathbf{H}_{\mathsf{msg}} \cdot \mathbf{G}$$
and that $\|\mathbf{U}\| \leq s \cdot \sqrt{m}$.
- $\mathsf{DAPS.Extract}(\mathsf{vk}, (\mathsf{subj}_1, \mathsf{payload}_1, \sigma_1), (\mathsf{subj}_2, \mathsf{payload}_2, \sigma_2))$: On input a verification key $\mathsf{vk} = (\mathbf{A}, \mathsf{ct})$, and two subject/message pairs $(\mathsf{subj}_1, \mathsf{payload}_1)$, $(\mathsf{subj}_2, \mathsf{payload}_2)$ and their signatures $\sigma_1 = \mathbf{U}_1, \sigma_2 = \mathbf{U}_2$, the extraction algorithm runs $\mathsf{sk}' \leftarrow \mathsf{PKE.Decrypt}(\mathbf{U}_1 - \mathbf{U}_2)$ and outputs sk'.

Signing Correctness. The correctness follows easily from the correctness of the trapdoor algorithm SamPre (Lemma 1) and the tail bounds of discrete Gaussian.

Security and Extrability. We now state the security and extractability of the construction above.

Theorem 2. *The* DAPS *construction above is unforgeable assuming the hardness of* $\mathrm{SIS}(n, m, q, \beta)$ *for* $\beta = O(s)$ *and circular security of* Π_{pke} *modeling the hash functions* $\mathsf{H}_{\mathsf{msg}}$ *and* $\mathsf{H}_{\mathsf{subj}}$ *as random oracles.*

Theorem 3. *Assuming* $\mathsf{H}_{\mathsf{msg}}$ *as a collision-resistant hash function, the* DAPS *construction above is* F_{DAPS}*-extractable.*

5 Extensions of DAPS to other Predicates

In this work, we provide sample PAPS constructions for a number of simple predicates. Due to space limitations, we describe and prove our constructions in the full version.

6 Multi-Authority Setting

For many practical situations, there is not a single authority signer, but multiple authorities who sign messages. For these type of situations, it is useful to extend the PAPS framework to the multi-authority setting where any compromising set of signatures by the different authorities reveals some private information of the signers.

We note that in this scenario, allowing a compromising set of signatures to reveal a secret key of a signer is not well-formulated in that any signer can compute a compromising set of signatures itself using its own signing key and extract another signer's secret key. Therefore, for the multi-authority setting, we let the extraction algorithm to reveal some predefined private data and define an additional algorithm $\mathsf{PAPS.CommGen}$ that takes in this private information that we denote by secret of a subset of the signers and generates a commitment comm pertaining to secret. For correctness, we require that a compromising set of signatures from these subset of signers along with the commitment comm allow anyone to reveal the private information secret.

Definition 9 (Multi-Authority PAPS). *A* multi-authority predicate-authentication-preventing signature *(MAPAPS) on a corresponding message space $\mathcal{M}$, secret space $\mathcal{S}$, and a predicate $f : \mathcal{M}^k \to \{0,1\}$ is a tuple of efficient algorithms consisting of the* PAPS *algorithms* $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.Extract})$ *with two additional algorithms* $(\mathsf{PAPS.CommGen}, \mathsf{PAPS.CommExtract})$ *defined as follows:*

- $\mathsf{PAPS.CommGen}(\{\mathsf{vk}_i\}, \mathsf{secret}) \to \mathsf{comm}_T$*: On input a set of verification keys $T = \{\mathsf{vk}_i\}$ and some private data $\mathsf{secret} \in \mathcal{S}$, the commitment generation algorithm generates a public commitment comm_T.*
- $\mathsf{PAPS.CommExtract}(\mathsf{comm}_T, \{(\mathsf{msg}_i, \sigma_i)\}_{i\in[k]}) \to \mathsf{secret}'$*: On input a public commitment comm_T, and a set of message and signature pairs, the extraction algorithm outputs private date secret'.*

Informally speaking, the $\mathsf{PAPS.CommGen}$ takes in a set of verification keys of the signers and generates a hiding commitment of some private data that belongs to these signers. On a compromising set of signatures produced by any of these signers allows anyone to extract this private information using the $\mathsf{PAPS.CommExtract}$ algorithm.

Correctness. As in the regular PAPS setting, we require that the algorithms $\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}$ satisfies the signing correctness requirement as in Section 3.

6.1 Extractability

In addition to the extractability property of $\mathsf{PAPS.Extract}$ in the single authority setting as in Section 3, we require an additional extractability in the multi-authority case where it is required that $\mathsf{PAPS.CommExtract}$ extract private information from the public commitment comm.

Definition 10. *Let $f : \mathcal{M}^k \to \{0,1\}$ be a predicate defined on k messages and $T = \{\mathsf{vk}_i\}$ be a set of verification keys. Then, a set of k message/signature pairs $\left\{(\mathsf{msg}_j, \sigma_j)\right\}_{j\in[k]}$ is* f-compromising *with respect to T if each signature σ_j is a valid signature of msg_j by some verification key $\mathsf{vk}_i \in T$, and the k messages satisfy the predicate f; that is, if for all $j \in [k]$, $\mathsf{PAPS.Verify}(\mathsf{vk}_i, \mathsf{msg}_j, \sigma_j) = 1$ for some $\mathsf{vk}_i \in T$ and $f(\mathsf{msg}_1, ..., \mathsf{msg}_k) = 1$.*

We define the standard extractability condition as follows.

Definition 11. *Fix a predicate $f : \mathcal{M}^k \to \{0,1\}$. We say that a* MAPAPS *scheme on a message space $\mathcal{M}$ is f-commitment-extractable if for all $\lambda \in N$, $\mathsf{secret} \in \mathcal{S}$, a set of verification keys $T = \{\mathsf{vk}_i\}$, $\mathsf{comm}_T \leftarrow \mathsf{PAPS.CommGen}(T, \mathsf{secret})$ and for all compromising set of signatures $S = \left\{(\mathsf{msg}_j, \sigma_j)\right\}_{j\in[k]}$ with respect to T, we have that $\mathsf{PAPS.CommExtract}(\mathsf{comm}_T, S) = \mathsf{secret}$ with overwhelming probability.*

6.2 Security

Commitment privacy. To prevent the extractability notion above from being satisfied trivially, we require a privacy requirement on the secret data of PAPS. Specifically, we require that an adversary with access limited to non-compromising sets of signatures do not learn information about the secret data. Let N be the number of authorities in the system.

Definition 12 (Privacy). *An f-extractable* PAPS *scheme $\mathit{\Pi}_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.CommGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.CommExtract})$ is data-hiding if for any efficient adversary $\mathcal{A}$, we have that*

$$\mathsf{Adv}^{\mathsf{dh}}_{\mathit{\Pi}_{\mathsf{paps}},\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \left|\Pr[\mathsf{Expt}^{(0)}_{\mathit{\Pi}_{\mathsf{paps}},\mathcal{A},\mathsf{dh}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\mathit{\Pi}_{\mathsf{paps}},\mathcal{A},\mathsf{dh}}(\lambda) = 1]\right| \le \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}^{(b)}_{\mathit{\Pi}_{\mathsf{paps}},\mathcal{A},\mathsf{dh}}$ is defined as follows:

1. $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{PAPS.KeyGen}(1^\lambda)$ *for $i = 1, ..., N$.*
2. $(T, \mathsf{secret}_0, \mathsf{secret}_1) \leftarrow \mathcal{A}_0(\{\mathsf{vk}\}_{i\in[N]})$.
3. $\mathsf{comm}_b \leftarrow \mathsf{PAPS.CommGen}(\{\mathsf{vk}_i\}_{t\in T}, \mathsf{secret}_b)$.
4. *Output* $\mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot)}(\{\mathsf{sk}_i\}_{i\in[N]\setminus T}, \mathsf{comm}_b)$.

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot)$ is defined as follows:

– *Oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot)$ maintains a list* SignedList *of all the previous valid queries made by $\mathcal{A}$. For a query msg and an index $i \in T$, that the adversary makes $\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot)$ checks whether there exists a compromising set of signatures with respect to T in* $\mathrm{SignedList} \cup \{\mathsf{msg}\}$. *If this is the case, the $\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot)$ outputs $\perp$. Otherwise, it outputs $\mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg})$.*

7 Multi-Authority DAPS

In this section we describe our construction of DAPS in the setting of multi-authority. We describe the scheme for the DAPS predicate, but the extensions in Section 5 translate directly to the multi authority setting since it uses the DAPS predicate as a black box. As in Section 4, we first describe a broadcast encryption scheme from the variant of the dual-Regev encryption scheme and then present our MADAPS construction using the broadcast encryption scheme. We prove that our scheme satisfies both the extractability and security requirements.

7.1 Broadcast Encryption

In this section, we present the broadcast encryption scheme. Again, the construction by itself is not interesting in that it follows generically from the dual-Regev public key encryption scheme in Section 4.1, but we present it separately for a more intuitive presentation. The scheme has the property that a trapdoor for any extended lattice that is defined by the concatenated public matrices allows for correct decryption. For clarity, we slightly alter the syntax of broadcast encryption where the global public key pk is divided into a number of public keys pk_i for each user in the system and the encryption algorithm takes in a subset of these public keys. Fix a security parameter n and let m, q, and χ be the corresponding parameter as in Section 4.1. We construct $\Pi_{\mathsf{be}} = (\mathsf{BE.KeyGen}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ as follows:

– $\mathsf{BE.KeyGen}(1^n, N)$: On input the security parameter 1^n, the key generation algorithm generates trapdoors $(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}i}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ for $i = 1, ..., N$. It outputs $\{(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}i})\}_{i\in[N]}$.

– $\mathsf{BE.Encrypt}(\{\mathsf{pk}_i\}_{i\in T}, m)$: On input a set of public keys $\{\mathsf{pk}_i\}_{i\in T} = \{\mathbf{A}_i\}_{i\in T}$ and a message $m \in \{0,1\}$, the encryption algorithm first samples uniformly random vectors $\mathbf{d}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and error vector $\mathbf{e}_0 \leftarrow \chi^m$ and then computes

$$\mathsf{ct}_0 = \mathbf{d}^T\mathbf{s} + \mathbf{e}_0 + \lceil q/2 \rceil \cdot m$$

Then for $t \in T$, it generates a fresh error vector from the noise distribution $\mathbf{e}_t \leftarrow \chi^m$ and computes

$$\mathsf{ct}_t = \mathbf{A}_t^T\mathbf{s} + \mathbf{e}_t$$

It outputs $\mathsf{ct}_T = (\mathsf{ct}_0, \{\mathsf{ct}_t\}_{t\in T})$.

– $\mathsf{BE.Decrypt}(\mathsf{sk}_i, \mathsf{ct}_T)$: On input the secret key sk_i for $i \in T$ and a ciphertext $\mathsf{ct}_T = (\mathsf{ct}_0, \{\mathsf{ct}_t\}_{t\in T})$, the decryption algorithm computes $\mathbf{s} \leftarrow \mathsf{Invert}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathsf{ct}_i)$. It then computes $m' = \mathsf{ct}_0 - \mathbf{d}^T\mathbf{s}$ and output $m \in \{0,1\}$ such that m' is closest to $\lceil q/2 \rceil \cdot m$.

We note that each ciphertext component $(\mathsf{ct}_0, \mathsf{ct}_i)$ makes up a regular ciphertext of the dual-Regev encryption scheme. Therefore, the correctness of the BE scheme above follows directly from the correctness of the dual-Regev scheme. Also, the construction of the scheme above is a generic concatenation of the regular public key encryption scheme and therefore, the security follows directly from the security of the public key encryption scheme.

As in the case of the dual-Regev public key encryption scheme, one can correctly decrypt the ciphertext even with a different trapdoor for the public matrices $\mathbf{A}_i$ as long as it is of sufficient quality. Furthermore, any trapdoor of a concatenated matrix $[\mathbf{A}_i \mid \mathbf{A}_j]$ is sufficient to recover the encryption randomness used in the encryption and therefore decrypt a ciphertext intended for users i and j. More formally, there exists a decryption algorithm as follows:

– $\mathsf{BE.Decrypt}'(\mathsf{sk}_{i,j}, \mathsf{ct}_T)$: On input a trapdoor $\mathsf{sk}_{i,j} = \mathbf{T}_{[\mathbf{A}_i|\mathbf{A}_j]}$ for $i, j \in T$, and a ciphertext $\mathsf{ct}_T = (\mathsf{ct}_0, \{\mathsf{ct}_t\}_{t \in T})$, the decryption algorithm computes $\mathbf{s} \leftarrow \mathsf{Invert}([\mathbf{A}_i|\mathbf{A}_j], \mathbf{T}_\mathbf{A}, [\mathsf{ct}_i|\mathsf{ct}_j])$. It then computes $m' = \mathsf{ct}_0 - \mathbf{d}^T\mathbf{s}$ and outputs $m \in \{0,1\}$ such that m' is closest to $\lceil q/2 \rceil \cdot m$.

7.2 Multi-Authority DAPS Construction

In this section, we extend the DAPS construction from Section 4.2 to the multi-authority setting (MADAPS). In addition to the algorithms in Π_{daps}, we define two additional algorithms DAPS.CommGen and DAPS.CommExtract as defined in Section 6. Let $\Pi_{\mathsf{be}} = (\mathsf{BE.KeyGen}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ be the broadcast encryption scheme as defined above with B-bounded noise distribution χ where $B = O(q/m^{3/2})$. Fix a security parameter $n \in \mathbb{N}$ and let m, q, s be the corresponding trapdoor parameters. We define the algorithms DAPS.CommGen and DAPS.CommExtract as follows:

– $\mathsf{DAPS.CommGen}(\{\mathsf{vk}_i\}, \mathsf{secret})$: On input a set of verification keys $T = \{\mathsf{vk}_i\} = \{\mathbf{A}_i\}$ and some private data $\mathsf{secret} \in \mathcal{S}$, the commitment generation algorithm computes $\mathsf{ct}_T \leftarrow \mathsf{BE.Encrypt}(T, \mathsf{secret})$ and sets $\mathsf{comm} = \mathsf{ct}_T$.
– $\mathsf{DAPS.CommExtract}(\mathsf{comm}_T, (\mathsf{subj}_0, \mathsf{payload}_0, \sigma_0), (\mathsf{subj}_1, \mathsf{payload}_1, \sigma_1))$: On input the parameters comm_T and a pair of compromising pair of signatures $\sigma_0 = \mathbf{U}_i$ and $\sigma_1 = \mathbf{U}_j$ corresponding to vk_i and vk_j respectively, the extraction algorithm concatenates the keys $\tilde{\mathbf{U}} = \begin{bmatrix} \mathbf{U}_i \\ -\mathbf{U}_j \end{bmatrix}$. It runs $\mathsf{secret}' \leftarrow \mathsf{BE.Decrypt}'(\tilde{\mathbf{U}}, \mathsf{comm}_T)$ and outputs the private data secret'.

Extractability and Privacy. We now state the extractability and privacy properties of the MADAPS construction above. We provide the proofs of the following theorems in the full version.

Theorem 4. *Assuming* $\mathsf{H}_{\mathsf{msg}}$ *is a collision-resistant hash function, the* MADAPS *construction above is* F_{DAPS}*-extractable.*

Theorem 5. *The* MADAPS *construction above is data-hiding assuming that the broadcast encryption scheme* Π_{be} *is secure.*

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h) ibe in the standard model. In *EUROCRYPT*. Springer, 2010.

ACPS09. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*. Springer, 2009.

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.

ALSY06. Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. Constant-size id-based linkable and revocable-iff-linked ring signature. In *INDOCRYPT*. Springer, 2006.

AP11. Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2011.

ASP12. Jacob Alperin-Sheriff and Chris Peikert. Circular and kdm security for identity-based encryption. In *PKC*. Springer, 2012.

BF14. Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *PKC*. Springer, 2014.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*. Springer, 2014.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*. ACM, 2012.

BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*. Springer, 2005.

BKM06. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC*. Springer, 2006.

BLP$^+$13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*. ACM, 2013.

BPS17. Mihir Bellare, Bertram Poettering, and Douglas Stebila. Deterring certificate subversion: efficient double-authentication-preventing signatures. In *IACR International Workshop on Public Key Cryptography*, pages 121–151. Springer, 2017.

BRS02. John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC*. Springer, 2002.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*. Springer, 2011.

BV14. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

BWZ14. Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In *CRYPTO*. Springer, 2014.

CD09. Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In *CRYPTO*. Springer, 2009.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*. Springer, 2001.

FN93. Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*. Springer, 1993.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*. ACM, 2008.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*. Springer, 2013.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*. ACM, 2015.

LLK15. Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, October 2015.

LM06. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*. Springer, 2006.

Lyu08. Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *PKC*. Springer, 2008.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*. Springer, 2012.

MBB$^+$15. Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: bringing key transparency to end users. In *USENIX Security 15*, pages 383–398, 2015.

Mic04. Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169, 2004.

MM11. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In *CRYPTO*. Springer, 2011.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*. Springer, 2012.

MP13. Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In *CRYPTO*. Springer, 2013.

MPR11. Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*. Springer, 2011.

MR07. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

Pei09. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*. ACM, 2009.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*. Springer, 2006.

PS14. Bertram Poettering and Douglas Stebila. Double-authentication-preventing signatures. In *ESORICS*. Springer, 2014.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

RST01. Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*. Springer, 2001.

TX03. Gene Tsudik and Shouhuai Xu. Accumulating composites and improved group signing. In *ASIACRYPT*. Springer, 2003.