Evaluating 2-DNF Formulas on Ciphertexts

Dan Boneh
dabo@cs.stanford.edu

Eu-Jin Goh
eujin@cs.stanford.edu

Kobbi Nissim
kobbi@cs.bgu.ac.il

April 2, 2006

Abstract

Let ψ be a 2-DNF formula on boolean variables $x_1, \ldots, x_n \in \{0,1\}$. We present a homomorphic public key encryption scheme that allows the public evaluation of ψ given an encryption of the variables $x_1, \ldots, x_n$. In other words, given the encryption of the bits $x_1, \ldots, x_n$, anyone can create the encryption of $\psi(x_1, \ldots, x_n)$. More generally, we can evaluate *quadratic* multi-variate polynomials on ciphertexts provided the resulting value falls within a small set. We present a number of applications of the system:

1. In a database of size n, the total communication in the basic step of the Kushilevitz-Ostrovsky PIR protocol is reduced from $\sqrt{n}$ to $\sqrt[3]{n}$.
2. An efficient election system based on homomorphic encryption where voters do not need to include non-interactive zero knowledge proofs that their ballots are valid. The election system is proved secure without random oracles but still efficient.
3. A protocol for universally verifiable computation.

1 Introduction

Secure computation allows several parties to compute a function of their joint inputs without revealing more than what is implied by their own inputs and the function outcome. Any polynomial time functionality can be computed by a secure protocol, requiring polynomial resources [36, 19]. These seminal results are obtained by a generic transformation that converts an insecure computation of a functionality to a secure version (often referred to as the 'garbled circuit' transformation).

Secure protocols generated from the garbled circuit transformation typically have poor efficiency. In particular, the communication complexity of the resulting protocols is proportional to the *size* of a circuit evaluating the functionality, and hence precludes sub-linear communication protocols. The result is that unless circuits are very small, the garbled circuit transformation is seldom used in protocols.

To avoid using the garbled circuit transformation, researchers have sought for tools that give more efficient protocols for specific functionalities. *Homomorphic encryption* enables "computing with encrypted data" and is hence a useful tool for secure protocols. Current homomorphic public key systems [20, 13, 28] have limited homomorphic properties: given two ciphertexts $\mathsf{Encrypt}(\mathcal{PK}, x)$ and $\mathsf{Encrypt}(\mathcal{PK}, y)$, anyone can compute either the sum $\mathsf{Encrypt}(\mathcal{PK}, x+y)$, or the product $\mathsf{Encrypt}(\mathcal{PK}, xy)$, but not both.[1] The problem of constructing 'doubly homomorphic' en-

[1] An exception is the scheme by Sander et al. [33] that is doubly homomorphic over a semigroup. On the other hand, the homomorphism comes with the cost of a constant factor expansion per semigroup operation. See also its comparison with our results in Section 1.1 below.

cryption schemes where one may both 'add and multiply' is a long standing open question already mentioned by Rivest et al. [32].

Homomorphic encryption schemes have many applications, such as protocols for electronic voting schemes [9, 2, 10, 11], computational private information retrieval (PIR) schemes [23], and private matching [15]. Systems with more general homomorphisms (such as both addition and multiplication) will benefit all these problems.

1.1 Our results

A homomorphic encryption scheme. We present a homomorphic public key encryption scheme based on finite groups of composite order that support a bilinear map. Using a construction along the lines of Paillier [28], we obtain a system with an additive homomorphism. In addition, the bilinear map allows for *one* multiplication on encrypted values. As a result, our system supports arbitrary additions and one multiplication (followed by arbitrary additions) on encrypted data. This property in turn allows the evaluation of multi-variate polynomials of total degree 2 on encrypted values. Our applications follow from this new capability.

The security of our scheme is based on a new hardness assumption that we put forward – the *subgroup decision problem.* Namely, given an element of a group of composite order $n = q_1 q_2$, it is infeasible to decide whether it belongs to a subgroup of order q_1.

Applications. As a direct application of the new homomorphic encryption scheme, we construct a protocol for obliviously evaluating 2-DNFs. Our protocol gives a quadratic improvement in communication complexity over garbled circuits. We show how to get a private information retrieval scheme (PIR) as a variant of the 2-DNF protocol. Our PIR scheme is based on that of Kushilevitz-Ostrovsky [23] and improves the total communication in the basic step of their PIR protocol from $\sqrt{n}$ to $\sqrt[3]{n}$ for a database of size n.

As noted above, our encryption scheme lets us evaluate quadratic multi-variate polynomials on ciphertexts provided the resulting value falls within a small set; in particular, we can compute dot products on ciphertexts. We use this property to create a gadget that enables the verification that an encrypted value is one of two 'good' values. We use this gadget to construct an efficient election protocol where voters do not need to provide proofs of vote validity. Finally, we generalize the election protocol to a protocol of universally verifiable computation.

Comparison to other public-key homomorphic systems. Most homomorphic systems provide only one homomorphism, either addition, multiplication, or xor. One exception is the system of Sander et al. [33] that provides the ability to evaluate NC^1 circuits on encrypted values. Clearly their construction also applies to 2-DNF formula. Unfortunately, the ciphertext length in their system grows exponentially in the depth of the 2-DNF formula when written using constant fan-in gates. In our system, the ciphertext size is independent of the formula size or depth; this property is essential for improving the communication complexity basic step of the Kushilevitz-Ostrovsky PIR protocol.

Organization. The rest of this paper is organized as follows. In Section 2 we review the bilinear groups underlying our construction and put forward our new hardness assumption. Section 3 details the construction of a semantically secure public key encryption scheme, its security and homomorphic properties. The basic application to 2-DNF evaluation is presented in Section 4, followed by the election and universally verifiable computation protocols in sections 5 and 6. Section 7 summarizes our results and poses some open problems.

2 Preliminaries

We briefly review the groups underlying our encryption scheme.

2.1 Bilinear groups

Our construction makes use of certain finite groups of composite order that support a bilinear map. We use the following notation:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of finite order n.
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. In other words, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$. We also require that $e(g, g)$ is a generator of $\mathbb{G}_1$.

We say that $\mathbb{G}$ is a bilinear group if there exists a group $\mathbb{G}_1$ and a bilinear map as above. In the next section we also add the requirement that the group action in $\mathbb{G}, \mathbb{G}_1$, and the bilinear map can be computed in polynomial time.

Constructing bilinear groups of a given order n. Let $n > 3$ be a given square-free integer that is not divisible by 3. We construct a bilinear group $\mathbb{G}$ of order n as follows:

1. Find the smallest positive integer $\ell \in \mathbb{Z}$ such that $p = \ell n - 1$ is prime and $p = 2 \bmod 3$.
2. Consider the group of points on the (super-singular) elliptic curve $y^2 = x^3 + 1$ defined over $\mathbb{F}_p$. Since $p = 2 \bmod 3$ the curve has $p + 1 = \ell n$ points in $\mathbb{F}_p$. Therefore the group of points on the curve has a subgroup of order n which we denote by $\mathbb{G}$.
3. Let $\mathbb{G}_1$ be the subgroup of $\mathbb{F}^*_{p^2}$ of order n. The modified Weil pairing on the curve [25, 22, 4, 26] gives a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the required properties.

2.2 The subgroup decision problem

We define an algorithm $\mathcal{G}$ that given a security parameter $\tau \in \mathbb{Z}^+$ outputs a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$ where $\mathbb{G}, \mathbb{G}_1$ are groups of order $n = q_1 q_2$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. On input τ, algorithm $\mathcal{G}$ works as follows:

1. Generate two random τ-bit primes q_1, q_2 and set $n = q_1 q_2 \in \mathbb{Z}$.
2. Generate a bilinear group $\mathbb{G}$ of order n as described at the end of Section 2.1. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be the bilinear map.
3. Output $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$.

We note that the group action in $\mathbb{G}, \mathbb{G}_1$ as well as the bilinear map can be computed in polynomial time in τ.

Let $\tau \in \mathbb{Z}^+$ and let $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$ be a tuple produced by $\mathcal{G}(\tau)$ where $n = q_1 q_2$. Consider the following problem: given $(n, \mathbb{G}, \mathbb{G}_1, e)$ and an element $x \in \mathbb{G}$, output '1' if the order of x is q_1 and output '0' otherwise; That is, without knowing the factorization of the group order n, decide if an element x is in a subgroup of $\mathbb{G}$. We refer to this problem as the *subgroup decision problem*. For an algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in solving the subgroup decision problem SD-Adv$_{\mathcal{A}}(\tau)$ is defined as:

$$\text{SD-Adv}_{\mathcal{A}}(\tau) = \left| \Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, x) = 1 \ : \ \begin{array}{c} (q_1, q_2, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(\tau), \\ n = q_1 q_2, \quad x \leftarrow \mathbb{G} \end{array} \right] \right.$$
$$\left. - \Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, x^{q_2}) = 1 \ : \ \begin{array}{c} (q_1, q_2, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(\tau), \\ n = q_1 q_2, \quad x \leftarrow \mathbb{G} \end{array} \right] \right|.$$

Definition 2.1. We say that $\mathcal{G}$ satisfies the subgroup decision assumption if for any polynomial time algorithm $\mathcal{A}$ we have that SD-Adv$_{\mathcal{A}}(\tau)$ is a negligible function in τ.

Informally, the assumption states that the uniform distribution on $\mathbb{G}$ is indistinguishable from the uniform distribution on a subgroup of $\mathbb{G}$. Recall that the factorization of the order of $\mathbb{G}$ is hidden so that the order of subgroups of $\mathbb{G}$ remains unknown to a polynomial time adversary.

3 A homomorphic public-key system

We can now describe our public key system. The system resembles the Paillier [28] and the Okamoto-Uchiyama [27] encryption schemes. We describe the three algorithms making up the system:

$\mathsf{KeyGen}(\tau)$**:** Given a security parameter $\tau \in \mathbb{Z}^+$, run $\mathcal{G}(\tau)$ to obtain a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$. Let $n = q_1 q_2$. Pick two random generators $g, u \stackrel{\text{R}}{\leftarrow} \mathbb{G}$ and set $h = u^{q_2}$. Then h is a random generator of the subgroup of $\mathbb{G}$ of order q_1. The public key is $\mathcal{PK} = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$. The private key is $\mathcal{SK} = q_1$.

$\mathsf{Encrypt}(\mathcal{PK}, M)$**:** We assume the message space consists of integers in the set $\{0, 1, \dots, T\}$ with $T < q_2$. We encrypt bits in our main application, in which case $T = 1$. To encrypt a message m using public key $\mathcal{PK}$, pick a random $r \stackrel{\text{R}}{\leftarrow} \{0, 1, \dots, n-1\}$ and compute

$$C = g^m h^r \in \mathbb{G}.$$

Output C as the ciphertext.

$\mathsf{Decrypt}(\mathcal{SK}, C)$**:** To decrypt a ciphertext C using the private key $\mathcal{SK} = q_1$, observe that

$$C^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m$$

Let $\hat{g} = g^{q_1}$. To recover m, it suffices to compute the discrete log of C^{q_1} base $\hat{g}$. Since $0 \leq m \leq T$ this takes expected time $\tilde{O}(\sqrt{T})$ using Pollard's lambda method [24, p.128].

Note that decryption in this system takes polynomial time in the size of the message space T. Therefore, the system as described above can only be used to encrypt short messages. Clearly one can use the system to encrypt longer messages, such as session keys, using any mode of operation that converts a cipher on a short block into a cipher on an arbitrary long block. We note that one can speed-up decryption by precomputing a (polynomial-size) table of powers of $\hat{g}$ so that decryption can occur in constant time.

3.1 Homomorphic properties

The system is clearly additively homomorphic. Let $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ be a public key. Given encryptions $C_1, C_2 \in G_1$ of messages $m_1, m_2 \in \{0, 1, \ldots, T\}$ respectively, anyone can create a uniformly distributed encryption of $m_1 + m_2 \bmod n$ by computing the product $C = C_1 C_2 h^r$ for a random r in $\{0, 1, \ldots, n-1\}$.

More importantly, anyone can multiply two encrypted messages *once* using the bilinear map. Set $g_1 = e(g, g)$ and $h_1 = e(g, h)$. Then g_1 is of order n and h_1 is of order q_1. Also, write $h = g^{\alpha q_2}$ for some (unknown) $\alpha \in \mathbb{Z}$. Suppose we are given two ciphertexts $C_1 = g^{m_1} h^{r_1} \in \mathbb{G}$ and $C_2 = g^{m_2} h^{r_2} \in \mathbb{G}$. To build an encryption of the product $m_1 \cdot m_2 \bmod n$ given only C_1 and C_2, do: 1) pick a random $r \in \mathbb{Z}_n$, and 2) set $C = e(C_1, C_2) h_1^r \in \mathbb{G}_1$. Then

$$\begin{aligned} C = e(C_1, C_2) h_1^r = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r &= g_1^{m_1 m_2} h_1^{m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r} \\ &= g_1^{m_1 m_2} h_1^{\tilde{r}} \in \mathbb{G}_1 \end{aligned}$$

where $\tilde{r} = m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r$ is distributed uniformly in $\mathbb{Z}_n$ as required. Thus, C is a uniformly distributed encryption of $m_1 m_2 \bmod n$, but in the group $\mathbb{G}_1$ rather than $\mathbb{G}$ (this is why we allow for just one multiplication). We note that the system is still additively homomorphic in $\mathbb{G}_1$.

Note. In some applications we avoid blinding with h^r, making the homomorphic computation deterministic.

Quadratic polynomials. Let $F(x_1, \ldots, x_u)$ be a u-variate polynomial of total degree 2. The discussion above shows that given the encryptions $C_1, \ldots, C_u$ of values $x_1, \ldots, x_u$, anyone can compute the encryption of $C = F(x_1, \ldots, x_u)$. On the other hand, to decrypt C, the decryptor must already know that the result $F(x_1, \ldots, x_u)$ lies in a certain polynomial size interval.

3.2 Security

We now turn to proving semantic security of the system under the subgroup decision assumption. The proof is standard and we briefly sketch it here.

Theorem 3.1. *The public key system of Section 3 is semantically secure assuming $\mathcal{G}$ satisfies the subgroup decision assumption.*

Proof. Suppose a polynomial time algorithm $\mathcal{B}$ breaks the semantic security of the system with advantage $\epsilon(\tau)$. We construct an algorithm $\mathcal{A}$ that breaks the subgroup decision assumption with the same advantage. Given $(n, \mathbb{G}, \mathbb{G}_1, e, x)$ as input, algorithm $\mathcal{A}$ works as follows:

1. $\mathcal{A}$ picks a random generator $g \in \mathbb{G}$ and gives algorithm $\mathcal{B}$ the public key $(n, \mathbb{G}, \mathbb{G}_1, e, g, x)$.
2. Algorithm $\mathcal{B}$ outputs two messages $m_0, m_1 \in \{0, 1, \ldots, T\}$ to which $\mathcal{A}$ responds with the ciphertext $C = g^{m_b} x^r \in \mathbb{G}$ for a random $b \xleftarrow{R} \{0, 1\}$ and random $r \xleftarrow{R} \{0, 1, \ldots, n-1\}$.
3. Algorithm $\mathcal{B}$ outputs its guess $b' \in \{0, 1\}$ for b. If $b = b'$ algorithm $\mathcal{A}$ outputs 1 (meaning x is uniform in a subgroup of $\mathbb{G}$); otherwise $\mathcal{A}$ outputs 0 (meaning x is uniform in $\mathbb{G}$).

It is easy to see that when x is uniform in $\mathbb{G}$, the challenge ciphertext C is uniformly distributed in $\mathbb{G}$ and is independent of the bit b. Hence, in this case $\Pr[b = b'] = 1/2$. On the other hand, when x is uniform in the q_1-subgroup of $\mathbb{G}$, then the public key and challenge C given to $\mathcal{B}$ are as in a real semantic security game. In this case, by the definition of $\mathcal{B}$, we know that $\Pr[b = b'] > 1/2 + \epsilon(\tau)$. It now follows that $\mathcal{A}$ satisfies SD-Adv$_{\mathcal{A}}(\tau) > \epsilon(\tau)$ and hence $\mathcal{A}$ breaks the subgroup decision assumption with advantage $\epsilon(\tau)$ as required. □

We note that if $\mathcal{G}$ satisfies the subgroup decision assumption then semantic security also holds for ciphertexts in $\mathbb{G}_1$. These ciphertexts are the output of the multiplicative homomorphism. If semantic security did not hold in $\mathbb{G}_1$, then it would also not hold in $\mathbb{G}$ because one can always translate a ciphertext in $\mathbb{G}$ to a ciphertext in $\mathbb{G}_1$ by "multiplying" by the encryption of 1. Hence, by Theorem 3.1, semantic security must also hold for ciphertexts in $\mathbb{G}_1$.

4 Two party efficient SFE for 2-DNF

In this section we show how to use our homomorphic encryption scheme to construct efficient secure function evaluation protocols. Our basic result is a direct application of the additive and multiplicative homomorphisms of our public key encryption scheme. We consider a two-party scenario where Alice holds a Boolean formula $\phi(x_1, \ldots, x_n)$ and Bob holds an assignment $a = a_1, \ldots, a_n$. As the outcome, Bob learns $\phi(a)$. We restrict our attention to 2-DNF formulas:

Definition 4.1. A 2-DNF formula over the variables $x_1, \ldots, x_n$ is of the form $\vee_{i=1}^{k} (\ell_{i,1} \wedge \ell_{i,2})$ where $\ell_{i,1}, \ell_{i,2} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$.

We first give a protocol for the model of semi-honest parties, and then modify it to cope with a malicious Bob, capitalizing on an 'input verification' gadget.

In the semi-honest model, both parties are assumed to perform computations and send messages according to their prescribed actions in the protocol. They may also record whatever they see during the protocol (i.e. their own input and randomness, and the messages they receive). On the other hand, a malicious party may deviate arbitrarily from the protocol. We sketch the security definitions for the simple case where only one party (Bob) is allowed to learn the output. We refer readers to Goldreich's book [18] for the complete definitions.

Security in the semi-honest model. The definition is straightforward since only one party (Bob) is allowed to learn the output:

- Bob's security – indistinguishability: We require that Alice cannot distinguish between the different possible inputs Bob may hold.
- Alice's security – comparison to an ideal model: Alice's security is formalized by considering an ideal trusted party that gets the inputs $\phi()$ and a, and gives $\phi(a)$ to Bob. We require in the real implementation that Bob does not get any information beyond whether a satisfies $\phi()$.

Security against malicious parties. The security definition for this model captures both the privacy and correctness of the protocol and is limited to the case where only one of the parties is corrupt. Informally, the security definition is based on a comparison with an ideal trusted party model (here the corrupt party may give an arbitrary input to the trusted functionality). The security requirement is that for any strategy a corrupt party may play in a real execution of the protocol, there is an efficient strategy it could play in the ideal model with computationally indistinguishable outcomes.

4.1 The basic protocol

Protocol `2-DNF` in Figure 1 uses our homomorphic encryption scheme for efficiently evaluating 2-DNFs with semi-honest parties. We get a three message protocol with communication complexity $O(n{\cdot}\tau)$ — a quadratic improvement in communication with respect to Yao's garbled-circuit protocol [36] that yields communication proportional to the potential formula length, $\Theta(n^2)$.

> INPUT: Alice holds a 2-DNF formula $\phi(x_1, \ldots, x_n) = \vee_{i=1}^{k} (\ell_{i,1} \wedge \ell_{i,2})$ and Bob holds an assignment $a = a_1, \ldots, a_n \in \{0,1\}^n$. Both parties' inputs include a security parameter τ.
>
> 1. Bob performs the following:
>   (a) He invokes $\mathsf{KeyGen}(\tau)$ to compute keys $\mathcal{SK}, \mathcal{PK}$, and sends $\mathcal{PK}$ to Alice.
>   (b) He computes and sends $\mathsf{Encrypt}(\mathcal{PK}, a_j)$ for $j = 1, \ldots, n$.
> 2. Alice performs the following:
>   (a) She computes an arithmetization Φ of ϕ by replacing "$\vee$" by "+", "$\wedge$" by "$\cdot$" and "$\bar{x}_j$" by "$(1 - x_j)$". Note that Φ is a polynomial in $x_1, \ldots, x_n$ with *total* degree 2.
>   (b) Alice computes the encryption of $r \cdot \Phi(a)$ for a randomly chosen r using the encryption scheme's homomorphic properties. The result is sent to Bob.
> 3. If Bob receives an encryption of 0, he outputs 0; otherwise, he outputs 1.

Figure 1: Protocol `2-DNF`.

Claim 4.1. *Protocol* `2-DNF` *is secure against semi-honest Alice and Bob.*

Proof Sketch. Alice's security follows as the distribution on Bob's output only depends on whether $\phi()$ is satisfied by a or not. Bob's security follows directly from the semantic security of the encryption scheme. □

Note. Protocol `2-DNF` (as well `Malicious-Bob-2-DNF` below) is secure even against a computationally unlimited Bob. Interestingly, the garbled circuit protocol (where Alice garbles ϕ) has the opposite property where it can be secured against an unbounded Alice but not an unbounded Bob. (See also Cachin et al. [6] for a discussion of computing on encrypted data versus garbled circuits).

4.2 Example application – private information retrieval

A private information retrieval (PIR) scheme allows a user to retrieve information from an n-bit database without revealing any information on which bit he is interested in [8, 23]. SPIR (symmetric PIR) is a PIR scheme that also protects the database privacy – a (semi-honest) user will only learn one of the database bits [17]. In this section, we show how an immediate application of protocol `2-DNF` results in a PIR/SPIR scheme. Our constructions are based on that of Kushilevitz and Ostrovsky [23].

A SPIR scheme. We get a SPIR scheme with communication $O(\tau \cdot \sqrt{n})$ as an immediate application of protocol `2-DNF`. Without loss of generality, we assume that the database size n is a perfect square and treat the database as a table D of dimensions $\sqrt{n} \times \sqrt{n}$. Using this notation, suppose

Bob wants to retrieve entry (I, J) of D. Alice (the database holder) holds the 2-DNF formula ϕ over $x_1, \ldots, x_{\sqrt{n}}, y_1, \ldots, y_{\sqrt{n}}$:

$$\phi(x_1, \ldots, x_{\sqrt{n}}, y_1, \ldots, y_{\sqrt{n}}) = \vee_{D_{i,j}=1} (x_i \wedge y_j),$$

and Bob's assignment a sets x_I and y_J to 1 and all other variables to 0. Bob and Alice carry out the `2-DNF` protocol with this assignment and 2-DNF formula. It is clear that $\phi(a) = D_{I,J}$ as required.

An alternative construction. Using the `2-DNF` protocol for SPIR restricts database entries to bits. We provide an alternative construction that allows each database entry to contain up to $O(\log n)$ bits. We consider the data as a table of dimensions $\sqrt{n} \times \sqrt{n}$ as above. To retrieve entry (I, J) of D, Bob creates two polynomials $p_1(x)$ and $p_2(x)$ of degree $\sqrt{n} - 1$ such that $p_1(i)$ is zero on $0 \le i < \sqrt{n}$ except for $p_1(I) = 1$, and similarly $p_2(j)$ is zero on $0 \le j < \sqrt{n}$ except for $p_2(J) = 1$. Bob sends to Alice the encryption of the coefficients of $p_1(x)$ and $p_2(x)$. Alice uses the encryption scheme's homomorphic properties to compute the encryption of

$$D_{I,J} = \sum_{0 \le i,j < \sqrt{n}} p_1(i) p_2(j) D_{i,j}.$$

We allow $D_{i,j}$ to be b-bit values where $b = O(\log n)$. Bob recovers $D_{i,j}$ in time $O(2^{b/2})$ by computing a discrete logarithm e.g. using the baby-step giant-step algorithm.

A PIR scheme. Standard communication balancing of our SPIR scheme results in a PIR scheme where each party sends $O(\tau \cdot \sqrt[3]{n})$ bits. In particular, view the database as comprising of $n^{1/3}$ chunks, each chunk containing $n^{2/3}$ entries, where Bob is interested in retrieving entry (I, J, K) of D. Bob sends Alice the coefficients of two polynomials $p_1(x)$ and $p_2(x)$ of degree $\sqrt[3]{n} - 1$ such that $p_1(i) = p_2(i) = 0$ on $0 \le i < \sqrt[3]{n}$ except for $p_1(I) = p_2(J) = 1$. Alice uses the encryption scheme's homomorphic properties to compute encryptions of

$$D_{I,J,k} = \sum_{0 \le i,j < \sqrt[3]{n}} p_1(i) p_2(j) D_{i,j,k}$$

for $0 \le k < \sqrt[3]{n}$. Alice sends the $\sqrt[3]{n}$ resulting ciphertexts to Bob who decrypts the Kth entry.

Recursively applying this balancing (as in Kushilevitz-Ostrovsky [23]) results in a protocol with communication complexity $O(\tau n^{\epsilon})$ for any $\epsilon > 0$. We note that the recursion depth to reach ϵ is lower in our case compared to that of Kushilevitz-Ostrovsky [23] by a constant factor of $\log_2 3$.

4.3 Security of the 2-DNF protocol against a malicious Bob.

A malicious Bob may try to learn about Alice's 2-DNF formula by sending Alice an encryption of a non-boolean assignment $a_1, \ldots, a_n$. He may also let Alice evaluate ϕ for an encrypted assignment that Bob cannot decrypt himself. Both types of behaviors do not correspond to a valid run in the ideal model.

To prevent the first attack, we present a gadget that allows Alice to ensure a ciphertext she receives contains one of two 'valid' messages v_0, v_1. This gadget is applicable outside of the scope of `2-DNF` as we demonstrate in sections 5 and 6. The second attack is prevented using standard methods — Alice presents Bob with a challenge that cannot be resolved unless he can decrypt.

This decryption ability is then used when Bob is simulated to create valid inputs for the trusted party.[2]

A gadget for checking $\mathbf{c} \in \{\mathbf{v_0}, \mathbf{v_1}\}$. This gadget exploits our ability to evaluate a polynomial of total degree 2 on the encryption of c. We choose a polynomial that has v_0 and v_1 as zeros as follows: given an encryption of a value c, Alice uses the homomorphic properties of the encryption scheme to compute $r \cdot (c - v_0) \cdot (c - v_1)$ for a randomly chosen r. For $c \in \{v_0, v_1\}$, this computation results in the encryption of 0. For other values of c, the result is random. In the special case of $c \in \{0, 1\}$, Alice computes $r \cdot c \cdot (c - 1)$.

The protocol. The result is protocol `Malicious-Bob-2-DNF` described in Figure 2.

INPUT: as in protocol `2-DNF` in Figure 1.

1. Bob invokes $\mathsf{KeyGen}(\tau)$ to compute keys $\mathcal{SK}, \mathcal{PK}$ and sends $\mathcal{PK} = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ to Alice.
2. Alice and Bob engage in a 'public key validation' protocol:
   (a) Bob proves to Alice in zero knowledge that n is a product of two primes [3, 35, 16, 7].
   (b) Alice checks that $g^n = h^n = 1 \in \mathbb{G}$ and $g, h \neq 1$. She rejects $\mathcal{PK}$ if not.
3. Next, Alice and Bob engage in a 'proof of decryption ability' protocol:
   (a) Alice chooses τ random bits $m_1, \ldots, m_\tau$ and sends to Bob all the encryptions $\mathsf{Encrypt}(\mathcal{PK}, m_1), \ldots, \mathsf{Encrypt}(\mathcal{PK}, m_\tau)$.
   (b) Bob replies with a decryption $m'_1, \ldots, m'_\tau$ of the received encryptions. Alice aborts if any of Bob's decryptions is incorrect.
4. Bob computes and sends $\mathsf{Encrypt}(\mathcal{PK}, a_j)$ for $j = 1, \ldots, n$.
5. Alice performs the following:
   (a) She computes an arithmetization Φ of ϕ as in protocol `2-DNF`.
   (b) Using the homomorphic properties of the encryption scheme, she computes the encryption of $r \cdot \Phi(a) + \sum_{i=1}^{n} r_i \cdot a_i \cdot (a_i - 1)$ for randomly chosen r, r_i. She sends the result to Bob.
6. If Bob receives an encryption of 0, he outputs 0; otherwise, he outputs 1.

Figure 2: Protocol `Malicious-Bob-2-DNF`.

Claim 4.2. *Protocol* `2-DNF` *is secure against semi-honest Alice and malicious Bob.*

Proof Sketch. Security against semi-honest Alice follows as in protocol `2-DNF`. Security against malicious Bob follows by simulation. Note that the 'proof of decryption ability' sub-protocol can be used to decrypt Bob's message in Step 4 of the protocol, hence providing the inputs to the trusted party. ∎

[2] The 'standard' use of this technique is to give Bob a random message for a challenge. Bob's simulator would then use the self reducibility properties of the encryption scheme to (i) map an encrypted message $\mathsf{Encrypt}(\mathcal{PK}, m)$ to an encryption of a random message, say $\mathsf{Encrypt}(\mathcal{PK}, m + r)$, (ii) use Bob's procedure to retrieve $m' = m + r$, and (iii) retrieve $m = m' - r$. As the message space is limited in our scheme due to decryption limitations, we need a slightly modified scheme.

5 An efficient election protocol without random oracles

In this section, we describe an electronic election protocol where voters submit boolean ("yes/no") votes. Such protocols were first considered by Benaloh and Fisher [9, 2] and more recently by Cramer et al. [10, 11].

A key component of electronic election schemes is a proof, attached to each vote, of its correctness (or validity); for example, a proof that the vote really is an encryption of 0 or 1. Otherwise, voters may corrupt the tally by sending an encryption of an arbitrary value. Such proofs of validity are typically zero-knowledge (or witness indistinguishable) proofs. These interactive zero knowledge proofs of bit encryption are efficiently constructed (using zero knowledge identification protocols) for standard homomorphic encryption schemes such as ElGamal [13, 21], Pedersen [29, 10], or Paillier [28, 12]. The proof of validity is then usually made non-interactive using the Fiat-Shamir heuristic of replacing communication with an access to a random oracle [14]. In the actual instantiation, the random oracle is replaced by some 'cryptographic function' such a hash function. Security is shown hence to hold in an ideal model with access to the random oracle, and not in the standard model [30].

Our election protocol has the interesting feature that voters do not need to include proofs of validity or any other information except for their encrypted votes when casting their ballots. Instead, the election authorities can jointly verify that a vote is valid based solely on its encryption. The technique is based on the gadget we constructed in Section 4.3. This gadget allows us to avoid using the Fiat-Shamir heuristic and yet makes our scheme efficient. As a result, our election scheme is very efficient from the voter's point of view as it requires only a single encryption operation (two exponentiations) to create a ballot.[3]

5.1 The election scheme

Our scheme belongs to the class of election protocols proposed by Cramer et al. [10, 11] where votes are encrypted using a homomorphic encryption scheme.

For robustness, we use a threshold version of the encryption scheme in Section 3. For simplicity (following Shoup [34]), we assume that a trusted dealer first generates the public/private keys, shares the private keys between the election authorities, and then deletes the private key (a generic secure computation may be used to replace the trusted dealer, as this is an offline phase). With this assumption, a threshold version of our encryption scheme can be constructed using standard techniques from discrete log threshold cryptosystems [29].

Correctness of threshold decryption. One caveat is that threshold decryption requires a zero knowledge of correct partial decryption from each election authority that contributes a share of its private key. Since the number of election authorities is typically a small constant, the proof of correct partial decryption can be performed interactively with relative efficiency between election authorities; transcripts of such interactions are made public for verification (note that transcripts do not leak information on votes). Another possible technique is to use a trusted source of random bits (such as a beacon [31]) among the election authorities, or for the authorities to collectively generate a public source of random bits. In a typical run of our protocol, the election authorities run only a limited number of these proofs (see below), hence the usage of either technique results

[3]Curiously, this voting scheme is probably the most efficient for the voter, taking into account the efficiency of operating in an elliptic curve group.

in a reasonably efficient protocol, and allows us to avoid using the Fiat-Shamir heuristic.

We note that these techniques can also be used in existing election protocols for verifying a voter's ballot, which avoids the Fiat-Shamir heuristic; but the resulting protocol becomes unwieldy and inefficient especially when the number of voters is large (and we expect that there is at least several orders of magnitude more voters than election authorities).

Vote verification. Here we use the verification gadget of Section 4.3 in combination with threshold decryption. We let all authorities compute an encryption of $v \cdot (v-1)$ and then jointly decrypt the result. To save on computation, we check a batch of votes at once (i.e. $\sum r_i \cdot v_i \cdot (v_i - 1)$ where the r_i's are chosen by the verifiers) and then run a binary search to identify the invalid votes [1].

The protocol. We assume the existence of an online bulletin board where the parties participating in the protocol post messages. Our election protocol works as follows:

Setup: As discussed above, a trusted dealer first generates the public parameters and the private key for the encryption scheme of Section 3, and shares the private key between the k election authorities so that at least t out the k election authorities are needed to decrypt. Finally, the trusted dealer deletes the private key and has no further role in the protocol. The public parameters are posted on a public bulletin board.

Denote one of the k election authorities as a leader (the election authority that organizes a quorum for decryption requests). After the public parameters are posted to the public board, the leader publishes an encryption of the bit 1 and the random bits used to create that encryption. Denote this encryption of the bit 1 as E_1. With the random bits, the other $k-1$ election authorities can check that E_1 is indeed an encryption of 1.

Vote casting: Voters cast their ballots by encrypting a bit indicating their vote, and then publishing the encrypted bit to the public bulletin board.

Vote verification: When a ballot v has been posted, all k election authorities compute a ciphertext c corresponding to $v \cdot (v-1)$ where E_1 is used as the encryption of "1" (hence, c is 'deterministic' given the encryption of v). The leader forms a quorum of $t-1$ other election authorities to decrypt c (the other election authorities agree to participate only if c agrees with the ciphertext they computed). If c decrypts to something other than 0, then the vote v is invalid and is discarded.

For better efficiency in optimistic scenarios, any number of votes $v_1, \dots, v_k$ can be verified in bulk by first computing $r_1 \cdot v_1 \cdot (v_1 - 1) + \ldots + r_k \cdot v_k \cdot (v_k - 1)$ where the r_i's are collectively chosen by the election authorities, and then checking that the decryption of the result is 0. All invalid votes are efficiently located by binary search. We note that in general it suffices for r_i to be relatively short, as the chance of $\sum r_i \cdot v_i \cdot (v_i - 1)$ being zero when some of the summed votes are invalid is exponentially small in $|r|$.

Vote tabulation and tally computation: After all the votes are posted and verified, all k election authorities each add all the valid encrypted votes on the public board together (using the additive homomorphic property of the encryption scheme) to form the tallied vote V. The leader obtains a quorum of election authorities to decrypt V. Each election authority decides whether to participate in the decryption request by comparing V with her own tally.

We note that our election protocol also possesses the necessary properties of voter privacy (from semantic security of the encryption scheme), universal verifiability (from the homomorphic property of the encryption scheme and also because all votes and proof transcripts are posted to the bulletin board), and robustness (from the threshold encryption scheme). The reader is referred to [9, 2, 10, 11] for discussions of these properties.

6 Universally Verifiable Computation

We now describe a related application for the gadget of Section 4.3. Consider an authority performing a computation, defined by a (publicly known) circuit C over the joint private inputs $a = (a_1, \ldots, a_n)$ of the n users. The authority publishes the outcome $C(a)$ in a way that 1) lets everyone check that the computation was performed correctly, but 2) does not reveal any other information on the private inputs. Besides voting, other applications of universally verifiable computation include auctions.

To simplify our presentation, we only consider the case where $a_i \in \{0, 1\}$; general inputs are treated similarly using any binary representation. We describe a single authority protocol that is easily transformed into a threshold multi-authority protocol using standard methods.

6.1 A protocol for verifying $C(a)$

Setup. The authority uses $\mathsf{KeyGen}(\tau)$ to generate a public-key/private-key pair $\mathcal{PK}, \mathcal{SK}$. We assume the existence of a bulletin board where each user i posts an encryption $c_i = \mathsf{Encrypt}(\mathcal{PK}, a_i)$ of her input. We also assume the existence of a random function H accessible by all parties, which implies that we prove security only in the random oracle model. As in the previous section, we can do without a random oracle in the multi-authority case (details omitted).

We first give a high level overview of the protocol. After all users post their encrypted inputs onto the bulletin board, the authority decrypts each user's input and evaluates the circuit on these inputs. In the process of evaluating the circuit, the authority computes and publishes ciphertexts for all the wire values in C. In addition, the authority also publishes an encryption of the bit 1 and the random bits used to create that encryption. Denote this encryption of the bit 1 as E_1. Finally, the authority publishes an encrypted value V and a witness-indistinguishable proof that V is an encryption of 0; for now, we defer the exact definition of V.

To convince a verifier that the circuit was computed correctly, the authority needs to prove that 1) all inputs are binary, and 2) all gate outputs are correct. We show how a verifier checks that both conditions hold using validators v that can be publicly constructed from the public encrypted wire values. We first show how to construct validators for the user inputs and gate outputs before showing how to use these validators to verify the computation.

Building Validators for User Inputs. In the process of evaluating the circuit, the authority publishes the values on every wire of the circuit. We enumerate the wires and denote the value on wire i as a_i. We also denote the encryption of a_i as c_i. Recall that each user posts $c_i = \mathsf{Encrypt}(\mathcal{PK}, a_i)$ of her input a_i on the bulletin board (the a_i's are never revealed in the clear). For each input wire i with value a_i, let $r_i = H(i, c_i)$; note that r_i can be computed by any of the parties. The validator for a_i is $v_i = r_i \cdot a_i \cdot (1 - a_i)$, and the computation occurs modulo q_2 where q_2 is one of the factors of the modulus n (recall that $\mathcal{SK} = q_1$). It is easy to see that the encryption of v_i can be computed by anyone given H, c_i, and E_1.

We note that even given q_2 and allowing a polynomial (in the security parameter τ) number of applications of H, the probability that an adversary successfully generates an invalid c_i with $v_i = 0$

is bounded by $O(\text{poly}(\tau)/q_2)$, which is negligible in τ.

Building Validators for Gate Outputs. Let $g \in C$ be a binary gate for which both input wires x, y are validated. In addition, let $G(x, y)$ be the bivariate polynomial of total degree 2 that realizes the gate g. For example, an AND gate has $G_{\text{AND}}(x, y) = xy$, an OR gate has $G_{\text{OR}}(x, y) = x+y-xy$, and a NOT gate has $G_{\text{NOT}}(x) = 1 - x$. The validator for the output wire (enumerated z) of gate g is $v_z = r_z \cdot (a_z - G(x, y))$ where $r_z = H(z, c_z)$ and c_z is the encryption of the value a_z on wire z. Again, it is easy to see that any party can compute the encryption of v_z given H, c_x, c_y, c_z, and E_1.

Verifying the Circuit Using Validators. Using the homomorphic properties of the encryption scheme, anyone can compute (by herself) an encryption of the sum of validators for the circuit. Note that if all posted encryptions are correct, then the sum of validators is zero. Otherwise, it is zero with only a negligible probability. The authority supplies its own version of the encrypted validator sum called V, together with a zero-knowledge proof that the resulting sum is zero; in this case, the encryption is of the form h^r so one can use protocols designed for the Pedersen encryption [29]. To verify the circuit computation, a verifier computes her own validator sum V', checks that $V' = V$, and then verifies the witness-indistinguishable proof that V is an encryption of 0.

7 Summary and open problems

We presented a homomorphic encryption scheme that supports addition and one multiplication. We require that the values being encrypted lie in a small range as is the case when encrypting bits. These homomorphic properties enable us to evaluate multi-variate polynomials of total degree 2 given the encrypted inputs. We described a number of applications of the system. Most notably, using our encryption scheme, we (i) reduced the amount of communication in the basic step of the Kushilevitz-Ostrovsky PIR, (ii) improved the efficiency of election systems based on homomorphic encryption, and (iii) implemented universally verifiable secure computation. We hope this scheme will have many other applications.

We end with a couple of open problems related to our encryption scheme:

n-linear maps. The multiplicative homomorphism was possible due to properties of bilinear maps. We note that an n-linear map would enable us to evaluate polynomials of total degree n rather than just quadratic polynomials. This provides yet another motivation for constructing cryptographic n-linear maps [5].

Message space. Our scheme is limited in the size of message space due to the need to compute discrete logarithms during decryption. An encryption scheme that allows for a large message space would enable more applications, such as an efficient shared RSA key generation.

Acknowledgements

We thank David Wagner for pointing out a mistake in the original 2-DNF protocol for malicious users.

References

[1] M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Proceedings of Eurocrypt '98*, volume 1403, 1998.

[2] J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, 1987.

[3] M. Blum. Coin flipping by telephone. In *Proceedings of Crypto '81*, 1981.

[4] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001.*

[5] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. In *Topics in Algebraic and Noncommutative Geometry*, number 324 in Contemporary Mathematics. American Mathematical Society, 2003.

[6] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. In *27th International Colloquium on Automata, Languages and Programming (ICALP '2000)*, volume 1853 of *Lecture Notes in Computer Science*, pages 512–523. Springer-Verlag, Berlin Germany, July 2000.

[7] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Eurocrypt '99*, pages 107–122, 1999.

[8] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, 23–25 Oct. 1995. IEEE.

[9] J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*, pages 372–382, 1985.

[10] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.

[11] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490, Sep 1997.

[12] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, 2001.

[13] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.

[14] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. Odlyzko, editor, *Proceedings of Crypto 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[15] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, May 2004.

[16] R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *ACM Computer and Communications Security (CCS) '98*, 1998.

[17] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, 2000.

[18] O. Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge Univesity Press, 2004.

[19] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187, Toronto, Ontario, Canada, 27–29 Oct. 1986. IEEE.

[20] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM Press, 1982.

[21] M. Jakobsson and A. Juels. Millimix: Mixing in small batches. Technical Report 99-33, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Oct 1999.

[22] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of 4th Algorithmic Number Theory Symposium*, number 1838 in LNCS, pages 385–394. Springer, Jul 2000.

[23] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, 20–22 Oct. 1997. IEEE.

[24] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[25] V. Miller. Short programs for functions on curves. Unpublished manuscript, 1986.

[26] V. Miller. The weil pairing, and its efficient calculation. *J. of Cryptology*, 17(4), 2004.

[27] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Proceedings of Eurocrypt 1998*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, May 1998.

[28] P. Pallier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, May 1999.

[29] T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. Davies, editor, *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 522–526. Springer, 1991.

[30] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[31] M. Rabin. Transaction protection by beacons. *Journal of Computer and System Science*, 27(2):256–267, 1983.

[32] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 1978.

[33] T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC^1. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*, pages 554–567, New York, NY, USA, Oct. 1999. IEEE Computer Society Press.

[34] V. Shoup. Practical threshold signatures. In B. Preneel, editor, *Proceedings of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000.

[35] J. van de Graaf and R. Peralta. A simple and secure way to show validity of your private key. In *Proceedings of Crypto '87*, 1987.

[36] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE Computer Society Press, 1982.