A PUBLIC-KEY THRESHOLD CRYPTOSYSTEM BASED ON RESIDUE RINGS

STEPHANIE DEACON, EDUARDO DUEÑEZ, AND JOSÉ IOVINO

Abstract. We present a generalization of Pedersen's public-key threshold cryptosystem. Pedersen's protocol relies on the field properties of $\mathbb{Z}_p$. We generalize the protocol so that the calculations can be performed in residue rings that are not necessarily fields. The protocol presented here is polynomial-time equivalent to Pedersen's.

Introduction

In [Ped91], Pedersen proposed a public-key threshold cryptosystem that does not require a trusted party for the generation of the private key. The calculations are performed in the field $\mathbb{Z}_p$, where p is a large prime. The field structure of $\mathbb{Z}_p$ is required for two aspects of the protocol: (i) the existence of discrete logarithms and (ii) the possibility of polynomial interpolation. The former provides the medium for secure communication, while the latter makes it possible for any set of individuals of a pre-established size to collaborate in order to generate the private key and use it for decryption without its becoming known to any of these individuals.

It is not difficult to see that in Pedersen's protocol the field $\mathbb{Z}_p$ can be replaced by any finite field. In this paper we generalize the protocol in a different direction: instead of $\mathbb{Z}_p$, the calculations are carried out in any finite ring of the form $\mathbb{Z}_N$, where $N = p^t$ or $N = 2p^t$. The restriction on N is necessary to ensure the existence of primitive roots (and hence discrete logarithms) modulo N. While Pedersen's protocol relies on the field structure to use polynomial interpolation, we show how this requirement can be removed by employing, for instance, a version of the Asmuth-Bloom secret sharing scheme [AB83].

Our protocol is polynomial-time equivalent, with respect to the exponent t, to Pedersen's.

The paper is organized as follows. In Section 1, we establish some preliminary number theoretic lemmas. In Section 2, we describe the process of selection and distribution of the keys. In Section 3, we show that the protocol indeed yields a threshold cryptosystem. The process of encryption and decryption is described in Section 4.

1. Preliminary Results

For real-valued functions f, g defined in some infinite interval (a, ∞), we write

$$f(x) \sim g(x) \quad \text{as } x \to \infty$$

1991 *Mathematics Subject Classification.* 68P25, 94A60, 11Y16, 11Z05.
Key words and phrases. Threshold cryptosystem, trusted party, residue rings.

if $\lim_{x\to\infty} f(x)/g(x) = 1$. For any real x, let $\pi(x)$ denote the number of primes no larger than x. The Prime Number Theorem (PNT) states that

$$\pi(x) \sim \frac{x}{\log x} \qquad \text{as } x \to \infty,$$

where the logarithm is natural. The following lemma is an immediate consequence of the PNT.

Lemma 1.1. *For any fixed real number κ strictly greater than 1,*

$$\pi(\kappa x) - \pi(x) \sim (\kappa - 1)\frac{x}{\log x} \qquad \text{as } x \to \infty. \tag{1}$$

Thus, for x large, there are plenty of primes in any interval of the form $(x, \kappa x]$; roughly, one in every $\log x$ integers in this interval will be a prime. Moreover, statements such as the following are true: if d is sufficiently large, about 90% of all primes having at most d (base 10) digits have exactly d digits. (To see why, apply Lemma 1.1 with $\kappa = 10$, $x = 10^{d-1}$. The difference on the left-hand side of equation (1) is equal to the number P_d of primes having exactly d digits (base 10). Hence, by Lemma 1.1, $P_d \sim 9 \cdot 10^{d-1}/\log(10^{d-1})$. By the PNT, the number $P_{\leq d}$ of primes having at most d digits (base 10) is $\pi(10^d) \sim 10^d/\log(10^d)$. Thus, $\frac{P_d}{P_{\leq d}} \sim 0.9\frac{d}{d-1} \sim 0.9$ as $d \to \infty$.)

Lemma 1.2. *Let N, n, and k be positive integers with $k \leq n$. Then, there exist pairwise relatively prime positive integers $m_1, m_2, \ldots, m_n$ such that*

(i) $\gcd(N, m_i) = 1$ *for* $i = 1, 2, \ldots, n$,
(ii) $N < m_1 < m_2 < \ldots < m_n$,
(iii) $\prod_{i=1}^{k} m_i > N \prod_{i=n-k+2}^{n} m_i$.

Moreover, $m_1, m_2, \ldots, m_n$ can be chosen to be prime numbers.

Proof. Fix a base $b > 1$, denote logarithms base b by $\log_b$, and assume that all numbers are written in base b. Any positive integer q has $\lfloor \log_b q \rfloor + 1$ digits (base b). Choose $d > \log_b N + k$ and distinct primes $m_1, \ldots, m_n$, each having d digits (base b). Then,

$$d - 1 \leq \log_b m_i < d, \tag{2}$$

and since $d - 1 > \log_b N + k - 1 > \log_b N$, it follows that $m_i > N$; thus, the primality of $m_1, \ldots, m_n$ implies (i). Changing indexes, if necessary, condition (ii) holds as well. As for condition (iii), upon taking logarithms (base b) of both sides, it is seen to be equivalent to

$$\sum_{i=1}^{k} \log_b m_i - \sum_{i=n-k+2}^{n} \log_b m_i > \log_b N. \tag{3}$$

By (2), the left-hand side of (3) is bounded below by $k(d-1) - (k-1)d = d - k$, so our choice of d ensures that condition (iii) holds, via inequality (3). □

Remark 1.1. An elementary counting argument shows that, in the limit as $X \to \infty$, if one chooses integers $m_1, m_2, \ldots, m_n \in [1, X]$ randomly (with uniform distribution), then the probability that they satisfy condition (i) above is strictly positive,

and equal to

$$\prod_{\substack{p \text{ prime} \\ p|N}} \left(1 + \sum_{\ell=1}^{n} (-1)^{\ell} p^{-\ell}\right) \prod_{\substack{p \text{ prime} \\ p\nmid N}} \left(1 - \sum_{\ell=2}^{n} (-1)^{\ell} (\ell - 1) \binom{n}{\ell} p^{-\ell}\right).$$

Since testing for relative primality can be done efficiently (using the Euclidean algorithm, for example), this observation suggests that a viable approach to finding the numbers $m_1, m_2, \ldots, m_n$ is simply to choose them at random in the range given by (2), above.

If N is a positive integer, $(\mathbb{Z}_N^*, \cdot)$ denotes the multiplicative group of units modulo N. This group is cyclic if and only if there exists $g \in \mathbb{Z}_N$ such that every $h \in \mathbb{Z}_N^*$ is of the form g^x for some integer x; such an element g is called a *primitive root* modulo N, or a *primitive root for* N.

We will use the following result from basic number theory.

Theorem 1.1. *The group $(\mathbb{Z}_N^*, \cdot)$ is cyclic if and only if N is 2 or 4, or N is of the form p^t, or $2p^t$, where p is an odd prime and t a positive integer.*

If g is a primitive root for N, x is an integer, and $h \equiv g^x \pmod{N}$, then x is called the discrete logarithm (base b and modulo N) of h. The multiplicative group $\mathbb{Z}_N^*$ has $\varphi(N)$ elements, where φ is Euler's totient function. By Euler's Theorem, $g^{\varphi(N)} \equiv 1 \pmod{N}$; therefore, discrete logarithms are only defined modulo $\varphi(N)$, since $g^x \equiv g^{x+\varphi(N)} \pmod{N}$.

For our purposes, N will henceforth be a fixed integer of the form p^t or $2p^t$ for some large prime p, which should be of a large enough size that finding discrete logarithms modulo p is computationally unfeasible.

It will be necessary to find and fix a primitive root modulo N. We assume that a primitive root g modulo p has already been found. This is feasible for primes p of cryptographically good size (say vis-à-vis Pedersen's original implementation). The following proposition, which is a complement to Theorem 1.1, shows that it is trivial to find a primitive root modulo N given one modulo p.

Proposition 1.1. *Let p be an odd prime, and let g be a primitive root modulo p. Then at least one of g and $g + p$ is a primitive root modulo p^2. Any primitive root modulo p^2 is also primitive modulo p^t for all t. Additionally, if h is a primitive root modulo p^t, then either h or $h + p^t$, whichever is odd, is a primitive root modulo $2p^t$.*

Proof. See, for example, Lemma 1.4.5 in [Coh93]. □

Remark 1.2. Proposition 1.1 remains true if stated with minus signs, i.e., with $g - p$, $h - p^t$ in place of $g + p$, $h + p^t$ (respectively).

Remark 1.3. As a note on implementation, one would apply Proposition 1.1 in the following fashion to produce a primitive root modulo N given a primitive root $\tilde{g}$ modulo p which one identifies with an integer $1 < \tilde{g} < p$: first "lift" $\tilde{g}$ to an integer $1 < g < N$ by choosing a random uniform integer $0 \leq k < N/p$ and setting $g = \tilde{g} + kp$. Then apply Proposition 1.1 to g. By applying the proposition with minus signs (see Remark 1.2 above), one can force the so-found primitive root modulo N to lie within the range $[1, N)$.

Remark 1.4. If one knows the discrete logarithm x to the base g of an integer h modulo N, then x (or, more properly, x modulo $p-1$) is itself the discrete logarithm

of h modulo p to the same base g. Hence, an encryption scheme whose security depends on the unfeasibility of finding discrete logarithms modulo N is no less secure than one depending on the same unfeasibility modulo p.

Regarding the converse to the preceding remark, computationally, it is not significantly more difficult to find discrete logarithms modulo N than modulo p; the following result shows that this can be done by successive approximations.

First notice that if g is a primitive root modulo p^t and $n < t$, then g is also primitive modulo p^n, hence $g^{\varphi(p^n)} \equiv 1 \pmod{p^n}$. Also, g is still primitive modulo p^{n+1}, so it has order $\varphi(p^{n+1}) \pmod{p^{n+1}}$, and hence $g^{\varphi(p^n)} \equiv 1 + \nu p^n \pmod{p^{n+1}}$ for some ν not divisible by p. Moreover, $g \pmod{p^{n+1}}$ determines $\nu \pmod{p}$ uniquely.

Proposition 1.2. *Let g be a primitive root modulo p^t, $h \in \mathbb{Z}_{p^t}^*$, and $x \in \mathbb{Z}$. Suppose that, for some $n < t$,*

$$g^x \equiv h \pmod{p^n} \quad \textit{and} \quad g^x \not\equiv h \pmod{p^{n+1}}.$$

Assume that $g^{\varphi(p^n)} \equiv 1+\nu p^n \pmod{p^{n+1}}$ with $\nu \in \mathbb{Z}_p^*$ and $g^x - h \equiv \lambda p^n \pmod{p^{n+1}}$ with $\lambda \in \mathbb{Z}_p^*$. If $\bar{h}, \bar{\nu}$ denote the multiplicative inverses of h, ν modulo p and

$$y = x - \bar{h}\bar{\nu}\lambda\varphi(p^n),$$

it holds that

$$g^y \equiv h \pmod{p^{n+1}}.$$

Proof. Let $\mu = -\bar{h}\bar{\nu}\lambda$. We have the following chain of congruences modulo p^{n+1} (the third line follows from the Binomial Theorem):

$$\begin{aligned} g^y &\equiv g^{x+\mu\varphi(p^n)} \\ &\equiv (h + \lambda p^n)(1 + \nu p^n)^\mu \\ &\equiv (h + \lambda p^n)(1 + \mu\nu p^n) \\ &\equiv h + (\lambda + h\mu\nu)p^n \pmod{p^{n+1}}. \end{aligned}$$

By definition of μ, $\lambda + h\mu\nu \equiv 0 \pmod{p}$, so the last term vanishes $\pmod{p^{n+1}}$ and $g^y \equiv h \pmod{p^{n+1}}$ as claimed. □

Repeated use (at most $t-1$ times) of the preceding lemma allows one to computationally efficiently "push up" a discrete logarithm modulo p to one modulo p^t. In the case that $N = 2p^t$ and g is a primitive root modulo N, g must be odd, as must h in Lemma 1.2. In this case, $g^x \equiv h \pmod{p^t}$ implies $g^x \equiv h \pmod{N}$.

Summarizing these findings:

Proposition 1.3. *The discrete logarithm problems modulo p and modulo a fixed N of the form p^t or $2p^t$ are equivalent, up to polynomial-time (in t and the number of bits of N) calculations.*

2. Selection and Distribution of the Keys

Consider a group of n members, denoted $P_1, P_2, \ldots, P_n$. They choose a positive integer $k \leq n$, which is the number of individuals who must work together to ascertain the private key. In addition, they must choose a large positive integer N of the form p^t or $2p^t$, where p is a large prime number and t is a positive integer, thereby fixing the ring $(\mathbb{Z}_N, +, \cdot)$ in which they will implement the algorithm. The members now choose a primitive root g modulo N. Also, the values $m_1, m_2, \ldots, m_n$

are chosen so as to satisfy conditions (i)–(iii) of Lemma 1.2 and M is defined by (5). The integers m_i are known by all n members. It will also be necessary to choose $T > t$ such that letting $\widetilde{N} = p^T$ we have

$$\varphi(\widetilde{N}) = (1 - p^{-1})\widetilde{N} > n \cdot m_n = n \max_i m_i. \tag{4}$$

By Proposition 1.1, we may assume that g is a primitive root modulo $\widetilde{N}$.

Define now

$$M = \prod_{i=1}^{k} m_i. \tag{5}$$

Since

$$M = \prod_{i=1}^{k} m_i > N \cdot \prod_{i=n-k+2}^{n} m_i$$

and $m_1 < m_2 < \ldots < m_n$, then for any permutation: $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$,

$$M \leq M_k^\pi \quad \text{and} \quad M > N M_{k-1}^\pi, \tag{6}$$

where

$$M_k^\pi = \prod_{i=1}^{k} m_{\pi(i)}, \qquad M_{k-1}^\pi = \prod_{i=1}^{k-1} m_{\pi(i)}.$$

2.1. **Selection and Computation of the Public and Private Keys.** For every $i = 1, 2, \ldots, n$:

- Member P_i secretly and randomly (with a uniform distribution) chooses an integer x_i such that $0 \leq x_i < \varphi(N)/n$.
- P_i computes $0 < h_i < N$ so $h_i \equiv g^{x_i} \pmod N$ but does not immediately reveal this value. Instead, he commits to h_i by converting it to binary form, obtaining a random string r_i, and computing his commitment $C(h_i, r_i)$. P_i then sends $C(h_i, r_i)$ to the other members.
- After every member has sent his commitment $C(h_j, r_j)$, P_i will disclose h_i to the other members. P_i will also verify that the values h_j disclosed to him are consistent with the commitments $C(h_j, r_j)$ he previously received.
- h is then computed as:

  $$h \equiv \prod_{i=1}^{n} h_i \pmod N.$$

  (It is irrelevant whether h is regarded as an integer or rather as a residue class modulo N.)
- Define the private key x to be*

  $$x = \sum_{i=1}^{n} x_i.$$

  Note that $0 \leq x < \varphi(N)$.

*While for encryption and decryption purposes x can be regarded as a residue class modulo $\varphi(N)$, it is important to note that here x is a specific non-negative integer. This will be crucial to the implementation of the key-sharing scheme.

- Define the public key as $(\widetilde{N}, N, g, h)$. Note that

$$g^x = g^{\sum_{i=1}^n x_i} = \prod_{i=1}^n g^{x_i} \equiv \prod_{i=1}^n h_i \equiv h \pmod{N}.$$

Remark 2.1. While the x_i are uniformly distributed on the interval $[0, \varphi(N)/n)$, the private key $x = \sum_{i=1}^n x_i$ is not uniformly distributed on $[0, \varphi(N))$ (as a random variable). Thus, a brute-force attacker who knows h and seeks to recover the private key x has some *a priori* information to direct his search: he will presumably check whether $g^y \equiv h \pmod{N}$ for values of y having the largest probability of being realized as values of the random variable x. Assuming that the cost (time or resources) of checking whether a particular y is the private key is independent of y (modulo $\varphi(N)$), then the amount of *a priori* information is measured by the *entropy* of the distribution measure of the random variable x, relative to the counting measure on the interval $0 \leq q < \varphi(N)$, $q \in \mathbb{Z}$. In the present context, if Z is a random variable taking values in a finite set I (in our case, $I = \{k : 0 \leq k < \varphi(N)\}$) and if $p_Z(i)$ is the probability that $Z = i$, then the entropy $S(Z)$ of Z is

$$S(Z) = -\sum_{i \in I} p_Z(i) \log p_Z(i),$$

with the convention that $0 \log 0 = 0$. The highest possible entropy is that of a uniform variable U on I (*i. e.*, $p_U(i) = |I|^{-1}$ for all $i \in I$), namely $S(U) = \log |I|$ (here $|I|$ is the number of elements in I). The lower the entropy, the higher the *a priori* information about the variable (otherwise said, entropy measures the "uncertainty" of a variable). If I consists of the integers in the interval $[0, \varphi(N))$, then the entropy of a uniformly distributed random variable on I is $\log \varphi(N)$ and one can roughly interpret this quantity as the number of digits (bits) of the number of attempts in a successful brute-force attack. For our choice of N, this entropy is about $t \log p$.[†]

How much information is gained from the non-uniformity of the random variable x? It is not hard to show that the entropy of x is about $t \log N - \frac{1}{2} \log n$. Hence, the lack of uniformity results in an *a priori* weakening of the system no worse than one due to (the number of digits of) the key space being reduced by $\frac{1}{2} \log n$.

Remark 2.2. The random bitstring r_i chosen above binds P_i to the integer that he sent out. We assume that the probability of any two members choosing the same random bitstring is small[‡]. The natural orderings of (i) the set of all m_i, and (ii) the set of chosen bitstrings will be used to determine which m_i corresponds to which member. Thus, for the rest of this protocol, the member corresponding to the random bitstring r_i will be known by the index i.

2.2. **Generation of the Shares.** Recall that the positive integer $k \leq n$, which is the number of individuals who must work together to ascertain x, has already been chosen by the members. All n members will follow the key-distribution protocol below. For $i = 1, 2, \ldots, n$:

[†]Note that, in this context, entropy only measures the cost of a brute-force attack. In particular, Proposition 1.3 provides enough information that an exhaustive search over a keyspace of size only $\varphi(p)$, plus an iterative procedure of length t, would suffice to find the private key.

[‡]Even if the bitstrings of P_i and P_j coincide, both P_i and P_j would realize it as soon as they use the other's commitment to verify the presently disclosed h_i, h_j.

- Member P_i secretly and randomly (with a uniform distribution) chooses an integer y_i such that $0 \le y_i < (M - \varphi(N))/(nN)$. Let $y = \sum_{i=1}^{n} y_i$. Then, $0 \le y < (M - \varphi(N))/N$ and

$$0 \le x + Ny < M. \tag{7}$$

- Member P_i computes

$$s_{ij} \equiv x_i + y_i N \pmod{m_j}, \qquad 0 \le s_{ij} < m_j, \tag{8}$$

  for $j = 1, 2, \ldots, n$.
- P_i computes and broadcasts $\sigma_{ij} \equiv g^{s_{ij}} \pmod{\widetilde{N}}$, $0 \le \sigma_{ij} < \widetilde{N}$, $j = 1, 2, \ldots, n$.
- P_i will wait until he has received the values σ_{tj} for $t = 1, 2, \ldots, n$, $t \ne i$, and $j = 1, 2, \ldots, n$. He will then sign and secretly send s_{ij} to P_j for $j = 1, 2, \ldots, n$. Note that P_i simply keeps s_{ii}.

  Remark 2.3. It is imperative that P_i not reveal the values secretly sent to him by the other members (namely s_{ji} for $j = 1, 2, \ldots, n$ and $i \ne j$) since these values are the information with which P_i will compute his share of the private key (see below).

- P_i computes his share s_i of the private key as the sum of the values that were secretly sent to him by the other members along with s_{ii}, namely,

$$s_i = \sum_{j=1}^{n} s_{ji}.$$

  Equations (4) and (8) ensure that $0 \le s_i < \varphi(\widetilde{N})$.
- P_i now must verify that the value s_{ji} that P_j secretly sent him is consistent with the values σ_{ji} that P_j broadcast to all the members; P_i must be sure that P_j did not mislead him. To verify this, P_i must simply confirm that

$$\begin{aligned} g^{s_i} = g^{\sum_{j=1}^{n} s_{ji}} &= \prod_{j=1}^{n} g^{s_{ji}} \\ &\equiv \sigma_{ji} \pmod{\widetilde{N}}. \end{aligned}$$

  If P_i finds that these values are inconsistent, he will immediately alert the others of the error and publish the values s_{ji} along with his signature.

Note that member P_i was able to verify that his share s_i of the private key is correct by verifying that the values s_{ji}, $j = 1, \ldots, n$ and $j \ne i$, secretly sent to him were consistent with the previously broadcast values σ_{ji}. However, the other members do not know if P_i received correct information or even if he computed his share correctly. They can verify this in the following way:

- P_i computes and sends $\sigma_i \equiv g^{s_i} \pmod{\widetilde{N}}$, $0 \le \sigma_i < \widetilde{N}$, to the other members.
- Member P_j confirms that

$$\begin{aligned} \sigma_i \equiv g^{s_i} = g^{\sum_{j=1}^{n} s_{ji}} &= \prod_{j=1}^{n} g^{s_{ji}} \\ &\equiv \prod_{j=1}^{n} \sigma_{ij} \pmod{\widetilde{N}}. \end{aligned}$$

Thus, all members in the group have verified not only that their share of the private key is correct, but also that the other members have computed their shares correctly. At this point, the computation and verification of the members' shares is finished.

3. Protocol Provides a Threshold Cryptosystem

By definition (Equation 8),

$$s_{ji} \equiv x_j + y_j N \pmod{m_i},$$

for $j = 1, 2, \ldots, n$. Thus,

$$\sum_{j=1}^{n} s_{ji} \equiv \sum_{j=1}^{n} (x_j + y_j N) = \sum_{j=1}^{n} x_j + N \sum_{j=1}^{n} y_j \pmod{m_i}. \tag{9}$$

Using the definitions of s_i, x and y, Equation (9) can be rewritten as:

$$s_i \equiv x + Ny \pmod{m_i}. \tag{10}$$

3.1. k Members Can Collaborate to Reveal the Private Key.

Consider a collusion C of k members $P_{\pi(1)}, P_{\pi(2)}, \ldots, P_{\pi(k)}$ of the group. The members of C construct the following system of congruences for the unknown Z:

$$\begin{aligned} Z &\equiv s_{\pi(1)} \pmod{m_{\pi(1)}} \\ Z &\equiv s_{\pi(2)} \pmod{m_{\pi(2)}} \\ &\vdots \\ Z &\equiv s_{\pi(k)} \pmod{m_{\pi(k)}}. \end{aligned} \tag{11}$$

Since the m_i are relatively prime, the Chinese Remainder Theorem ensures that this system has a solution which is unique modulo M_k^π (and can be efficiently computed). By Equations (6) and (7), we have $0 \leq x + Ny < M \leq M_k^\pi$. Then, by (10), $Z = z = x + Ny$ is the only solution to (11) with $0 \leq Z < \prod_j m_{\pi(j)}$. The k members then solve for X in

$$X \equiv z \pmod{N}$$

finding a unique solution $X = x$ with $0 \leq X < N$ (recall that $0 \leq x < N$ by construction). Hence, k members can work together in a collusion to discover the private key.

3.2. Fewer Than k Members Cannot Find the Private Key.

Suppose that any $k-1$ members, $P_{\pi(1)}, P_{\pi(2)}, \ldots, P_{\pi(k-1)}$, form a collusion C, pool their shares $(s_{\pi(i)}, m_{\pi(i)})$, and attempt to find the private key x. We will show that these $k-1$ members will be unsuccessful and will indeed gain essentially no information about x.

Again, recall from (10) that $s_{\pi(i)} \equiv x + Ny \pmod{m_{\pi(i)}}$ for $i = 1, 2, \ldots, k-1$. Analogously to what was done in the previous section, the members of C may solve the system (11) (with $k-1$ instead of k) in order to gain knowledge about the $x + Ny$ and, ultimately, about x itself. Let us now show that this information holds little value for the members of C.

Write $x_c = \sum_i x_{\pi(i)}$, $x_{\bar{c}} = x - x_c$, and similarly for y_c and $y_{\bar{c}}$. Also let $w_c = x_c + Ny_c$ and $w_{\bar{c}} = x_{\bar{c}} + Ny_{\bar{c}}$. Note that

(12)

$$\begin{array}{lll} 0 \le x_c < \dfrac{k}{n}\varphi(N) & 0 \le y_c < \dfrac{k}{n}\dfrac{M-\varphi(N)}{N} & 0 \le w_c < \dfrac{k}{n}M \\ 0 \le x_{\bar{c}} < \left(1-\dfrac{k}{n}\right)\varphi(N) & 0 \le y_{\bar{c}} < \left(1-\dfrac{k}{n}\right)\dfrac{M-\varphi(N)}{N} & 0 \le w_{\bar{c}} < \left(1-\dfrac{k}{n}\right)M \end{array}$$

Those variables with a subscript 'c' are known to the collusion C, whereas those with a subscript '$\bar{c}$' are not. Moreover, the system of congruences

$$\begin{aligned} Z &\equiv s_{\pi(1)} - w_c \pmod{m_{\pi(1)}} \\ Z &\equiv s_{\pi(2)} - w_c \pmod{m_{\pi(2)}} \\ &\vdots \\ Z &\equiv s_{\pi(k-1)} - w_c \pmod{m_{\pi(k-1)}} \end{aligned} \tag{13}$$

is solved by $w_{\bar{c}} \pmod{M^{\pi}_{k-1}}$. However, note that $M/M^{\pi}_{k-1} > N$ by (6) so, in principle, any solution $Z = z$ of (13), say, the one with $0 \le z < M^{\pi}_{k-1}$, can be lifted to roughly $(1-k/n)M/M^{\pi}_{k-1} > (1-k/n)N$ distinct candidate values for $w_{\bar{c}}$. Hence, a brute-force attack is quite unfeasible.

A slightly more refined argument goes as follows. The entropy of $x_{\bar{c}}$ can be shown to be about $\frac{1}{2}\log(n-k) - \log n + \log N$ and that of $y_{\bar{c}}$ about $\frac{1}{2}\log(n-k) - \log n + \log M - \log N$. In the case at hand, the entropy of $w_{\bar{c}} = x_{\bar{c}} + Ny_{\bar{c}}$ is equal to the sum of the entropies of $x_{\bar{c}}$ and $y_{\bar{c}}$ (because the latter are independent, and there is a bijection $(x_{\bar{c}}, y_{\bar{c}}) \leftrightarrow w_{\bar{c}}$). In conclusion, the entropy of $w_{\bar{c}}$ is about $\log M - 2\log n + \log(n-k)$, hence practically as large as that of a uniform variable in the range $0 \le W < M$: the *a priori* information gained by C amounts to no more than about $2\log n$ bits.

Remark 3.1. Because the private key can be found by a collusion of k members, this cryptosystem assumes that at most $k-1$ members are dishonest. If k dishonest members cheat, instead of giving their shares to the trusted party, they could find and keep x for themselves.

4. Encryption and Decryption

Assume that a group of n members, $P_1, P_2, \ldots, P_n$, has followed the above key selection protocol (Section 2.1) to generate a group public key $(\widetilde{N}, N, g, h)$. In addition, each member computed his share of the private key via the key distribution protocol set forth in Section 2.2; thus member P_i holds the ordered pair (s_i, m_i) for $i = 1, 2, \ldots, n$.

Suppose that person B wants to send the group a message $Q \in \mathbb{Z}_N$. First, he will acquire the group's public key $(\widetilde{N}, N, g, h)$. He will then obtain a random number $1 \le l \le \varphi(N) - 1$, compute $\gamma \equiv g^l \pmod{N}$ and $\delta \equiv Q \cdot h^l \pmod{N}$, and send the ordered pair (γ, δ) to the group as the ciphertext.

To decrypt (γ, δ), no less than k members must send their shares (s_i, m_i) along with the ciphertext (γ, δ) to a trusted party (*e. g.*, a central server). The trusted party will use the Chinese Remainder Theorem to extract x from the k shares (Section 3.1). He will then use the private key x to decrypt the ciphertext (γ, δ) by computing $M \equiv (\gamma^x)^{-1} \cdot \delta \pmod{N}$ and will send the plaintext message M to the

k members of the group who sent their shares. The trusted party will then destroy these shares and the private key x. In this way, the group can continue to use the same key. Then, the next time the group is sent a message, any k members can again send their shares to the trusted party and receive the plaintext from him.

References

[AB83] Charles Asmuth and John Bloom. A modular approach to key safeguarding. *IEEE Trans. Inform. Theory*, 29(2):208–210, 1983.

[Coh93] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology - proceedings of EUROCRYPT '91*, Lecture Notes in Computer Science, pages 522–526, 1991.

Tesoro Corporation, 300 Concord Plaza, San Antonio, TX 78216-6999, USA
E-mail address, Stephanie Deacon: sdeacon@tsocorp.com

Department of Mathematics, The University of Texas at San Antonio, San Antonio, TX 78249, USA
E-mail address, Eduardo Dueñez: eduenez@math.utsa.edu

Department of Mathematics, The University of Texas at San Antonio, San Antonio, TX 78249, USA
E-mail address, José Iovino: iovino@math.utsa.edu