Binding ElGamal: a fraud-detectable alternative to key-escrow proposals

Eric R. Verheul*[1] and Henk C.A. van Tilborg[2]

[1] Ministry of the Interior, P.O. Box 20010, 2500 EA, The Hague, The Netherlands. Eric.Verheul@pobox.com

[2] Department of Math. and Comp. Sc., P.O. Box 513, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands. henkvt@win.tue.nl

Abstract. We propose a concept for a worldwide information security infrastructure that protects law-abiding citizens, but not criminals, even if the latter use it fraudulently (i.e. when not complying with the agreed rules). It can be seen as a middle course between the inflexible but fraud-resistant KMI-proposal [8] and the flexible but non-fraud-resistant concept used in TIS-CKE [2]. Our concept consists of adding *binding data* to the latter concept, which will not *prevent* fraud by criminals but makes it at least *detectable* by third parties without the need of any secret information. In [19], we depict a worldwide framework in which this concept could present a security tool that is flexible enough to be incorporated in any national cryptography policy, on both the domestic and foreign use of cryptography. Here, we present a construction for binding data for ElGamal type public key encryption schemes. As a side result we show that a particular simplification in a multiuser version of ElGamal does not affect its security.

Key words ElGamal, Traceable ElGamal, Key Escrow, Key Recovery

1 Introduction

We'll briefly summarize the *technical* position taken in [19]. A robust, worldwide information security infrastructure (ISI) must be set up which includes a Key Management Infrastructure which will (likely) be based on public key cryptography. Proper certification of public keys will be a crucial (and elaborate) service within this ISI. However, the unconditional use of encryption by criminals poses a threat to law enforcement, a problem that is hard to solve. Consequently, most governments feel that they have to realize two tasks. The first is to stimulate the establishment of an ISI which protects the legitimate interests of all relevant parties (businesses, governments, citizens), but which does not aid criminals. The second task is to cope with the use of other encryption techniques by criminals. How to achieve the second goal is outside the scope of this contribution, but it is our feeling that an ISI, that is widely accepted and trusted, will make it

* Views expressed here are personal and not necessarily shared by my employer.

easier to achieve the second task. We also feel that without strong cooperation of governments such a widely accepted and trusted ISI will never be established at all. In this paper we address a construction of a reliable ISI, which does not aid criminals.

In public key encryption (pke) encrypted messages - ideally - consist of two components:

C1. The (actual) message M encrypted with a symmetric system, using a random session key S.
C2. The session key S encrypted using the public key(s) of the addressee(s).

A straightforward method to prevent facilitation of criminals is outlined in the U.S.-government (draft) Key Management Infrastructure (KMI) proposal [8]. Here, participating users have to deposit their private keys with a private-sector Trusted Recovery Party (TRP).[3] When a law-enforcement agency (LEA), that has obtained legal authority to access a user's communication, strikes upon data encrypted within this scheme, the TRP will "relinquish information sufficient to access" these data. One of the problems mentioned in [19], is that the scheme is inflexible in an international context: in order to let the principle work for *any* country, *every* participating country - irrespective of its national policy on cryptography - has to escrow the private keys of its users also. Also, international cooperation of a TRP with a LEA outside the country of the TRP might be difficult and time-consuming. Although the latter problem is resolved in the "Royal Holloway" variant [11] of this scheme, it can be argued that the resulting flexibility here is not better than that of the KMI-proposal. Compare [1].

A more flexible method to prevent facilitation of criminals consists of *virtual addressing* session keys to Trusted Recovery Parties (see, for instance, the TIS Commercial Key Escrow [2]). In this scheme, participating users agree to add a third component to an encrypted message:

C3. The same session key S encrypted using the public key(s) of one or more Trusted Recovery Parties.

In effect, any TRP is treated as a virtual addressee, although the message is not sent to it. When a LEA is conducting a lawful intercept and strikes upon an enciphered message, they take the information component of one of the TRP's to that TRP. If shown an appropriate warrant, the TRP decrypts ("recovers") the information component and (only) hands over the session key S, so that the LEA agency can access the message.
This concept has been the base of several escrow products (Translucent Cryptography, AT&T Crypto Backup, RSA secure). Observe that users do not have to deposit secret key information to TRP's *beforehand*. This makes this approach

[3] We use the notion "Trusted Recovery Party" as it forms a combination of the (recent) U.S. notion "recovery" (replacing "key-escrow") and the European notion "Trusted Third Party".

more feasible (and acceptable to users) than the KMI-proposal; an important advantage as - also pointed out in the study of the National Research Council (NRC) [14, p.329] - feasibility of key-recovery solutions is a significant issue. We remark that one could incorporate information in the session key identifying the sender (as is done in TIS-CKE). However, as this, in principle, makes possible a (partially) known-plaintext attack (cf. [4]) one should be careful with this.
Although this concept is very flexible (see below), its main drawback is that it offers no possibility, at least for others than the TRP, to check whether the third component actually contains the (right) session key; moreover the TRP can only discover "fraud" (i.e. not complying with the agreement) after a lawful wiretap. Hence, by sending noise instead of a third component unilateral abuse (i.e. without help of the addressee) is easily possible. This can be prevented in the software of the addressee by a recalculation and validation of C3 prior to decryption. However, abuse by colluding of a sender and receiver - through a one-time manipulation of this validation in software - is still easily possible. So the solution is almost entirely unenforceable. According to the NRC-study [14, p.214] U.S. senior Administration officials have said that this matter is the reason for the limitation to (only) 64 bits in the (draft) 1995 U.S. Key Escrow Export Criteria for cryptographic applications in software: "the limitation to 64 bits is a way of hedging against the possibility of finding easily proliferated ways to break the escrow binding built into software, with that result that U.S. software products without effective key escrow would become available worldwide". On the other hand, it is noted in the NRC-study [14, p.211] that a recovery encryption product does not have to be perfectly resistant to breaking the recovery binding: it should only be more difficult to bypass the recovery features than to build a system without recovery.

In [19] we looked for a middle course between the inflexible but fraud-resistant KMI-proposal and the flexible but non-fraud-resistant virtual addressing. We found one by not *preventing* colluding of sender and receiver, but by making it at least *detectable* by third parties without having access to secret (key) information. More specifically, we proposed the *binding alternative*, which adds a fourth component to the encrypted message:

C4. Binding data.

The idea is that any (third party) monitor, e.g., a network or (Internet) service provider, who has access to components C2, C3, and C4 (but not to any additional secret information) can determine that the session keys encrypted in components C2 and C3 coincide but it can not determine any information on the actual session key S. In this way, fraud is easily detectable (and punishable). Metaphorically speaking, binding data consists of equipping public-key encryption schemes used for confidentiality with a metal detector, as used at boarding gates on airports.
The binding concept supports the virtual addressing of session keys to several TRP's (or none for that matter), for instance, one to a TRP in the country of the sender S and one in the country of the addressee A. Note that this can be easily

implemented: S's software can (once) be adjusted to the public key of S's TRP; the public key of A's TRP can be part of A's (certified) public key. The solution therefore offers the same advantage for worldwide usability as [11]. We also remark that the binding concept also supports the functionality of controllable key splitting in the sense of Micali [13], even in several fashions. For instance, the private TRP key can be splitted in several parts and be deposited at several sub-TRP's. It turns out that the ElGamal system very conveniently supports the splitting and the reconstruction of private keys (see the end of Section 2). Finally, we remark that the time-boundedness condition (cf. [12, p.199]), i.e. the condition that time-limits on warrants can be enforced, can be fulfilled by additionally demanding that encrypted information (or all components) be time-stamped and signed by the sender. These can be easily verified by any third party monitor as well. A much simpler solution is to let the time be an (unencrypted) part of the message and to incorporate it in the binding data (as indicated in Section 4).

An additional feature could prevent the threat of the "tempted policemen" This tempted policemen might conspire with a criminal and have the criminal resent (or "receive") an unrelated, highly confidential business message intercepted by the policemen. The TRP, thinking the message originated from the (wiretapped) criminal, would assist the policemen in decrypting. In the binding scheme, this can be prevented by additionally requiring senders to virtually address the session key to themselves as well. The TRP could check this component before assisting a law-enforcement agency, and monitors could check on compliance. Incidentally, this feature can also solve similar problems in TIS-CKE and in the U.S. KMI-proposal. In the latter, it also overcomes the problem of international communications: the TRP has got the private key of the sender and can therefore recover the session key. Thus, binding cryptography can also benefit other proposals.

In [19], we depict a general framework in which the binding concept (as general notion) could present a security tool that is flexible enough to be incorporated in any national cryptography policy, for both the domestic and foreign use of cryptography, and that offers a flexible choice of trust for users. Here, we present a construction for binding data for the ElGamal type of pke schemes; this is particularly interesting as on 29 April 1997, ElGamal will no longer be encumbered by patents in the U.S..

A difficulty one faces in the construction of binding data for a pke scheme, apart from the binding data itself, is finding a suitable multiuser extension of it, allowing the secure (!) encryption of exactly the same session-key (i.e. including "padding" data) with different public keys. For the RSA scheme, for instance, this presents a problem (cf. [10]). In Section 2 we will introduce a secure multi-user extension of ElGamal. Section 3 deals with proving knowledge of equality of certain logarithmic values. Section 4 presents the construction of binding data techniques for ElGamal's protocol. Finally, many of the constructions for the ElGamal scheme can be extended to Desmedt's traceable variant of ElGamal ([6]). We will sketch some of these extensions in Appendix B.

2 The Multiuser ElGamal Encryption Scheme

The ElGamal [7] pke system makes use of a subgroup G of a multiplicative, cyclic group H in which the discrete logarithm problem is intractable. Let q be the order of G and let g be a generator of G. The elements g, G, and H are given to all participants by an Issuing Party (IP). We will not further specify G, H, but in a typical example H is the multiplicative group of Z/pZ for a (large) prime p and $G = H$.
To participate in the system, each participant P chooses his own secret key x_P (a random number less than q) and publishes his (certified) public key $y_P = g^{x_P} \in G$. If a person, say Ann, wants to encrypt a message $S \in H$ meant for participant Bob, she chooses a random number k less than q and sends the pair $(t, u) = (g^k, y_{Bob}^k \cdot S)$ to Bob. When Bob receives (t, u) he just calculates $u/t^{x_{Bob}}$ to find S back.
We focus on the following multiuser extension of ElGamal,

Definition 2.1 *In the* Multi-ElGamal *protocol, participant P, when going to encrypt message $S \in H$ for n participants with public ElGamal keys $y_1, y_2, \ldots, y_n$, will generate a random number k less than q and send pair $(g^k, y_i^k \cdot S)$ to the i-th participant,* $1 \leq i \leq n$.

The question that arises of course is whether Multi-ElGamal is less secure than choosing a different k for each participant (which is less efficient). We shall show it is not.
The following terminology is convenient. Let g be an element of G, y an element of the cyclic group $< g >$ generated by g, $S \in H$ and $k \in Z/qZ$. Then the 4-tuple $(g, y, g^k, y^k \cdot S)$ is called an *encryption* of g, y, k, S and will be denoted by $[g, y, k, S]$. The elements $k, S, \log_g y$ will be called the *secret* (or *unknown*) components of the encryption.

Lemma 2.2 *Let $[g, y_P, k_i, S_i]$, $1 \leq i \leq h$, be a sequence ("history") of encryptions for user P. Then anyone can construct a second sequence of encryptions $[g, \hat{y}, k_i, S_i]$, $1 \leq i \leq h$, with $\hat{y}$ random in G (but with the same k_i's and S_i's) such that the computation of $\log_g(y_P)$ is as difficult as that of $\log_g \hat{y}$.*

Proof: For $i = 1, 2, \ldots, h$, denote $(g^{k_i}, {y_P}^{k_i} \cdot S_i)$ by (A_i, B_i). Let i be one of $1, 2, \ldots, h$. Choose j randomly in Z/qZ, and compute $C = g^j$, $D_i = (A_i)^j$ and $\hat{y} = y_P \cdot C$. First of all, we observe that $\hat{y} = g^{x_P + j}$. So $\hat{y}$ is a random element in $< g >= G$.
Now $(g, \hat{y}, A_i, B_i \cdot D_i)$ can be computed. We shall prove that it is indeed an encryption $[g, \hat{y}, k_i, S_i]$. To this end the only condition that needs to be verified is $B_i \cdot D_i = \hat{y}^{k_i} \cdot S_i$. This follows from:

$$B_i \cdot D_i = y_P^{k_i} \cdot S_i \cdot g^{j \cdot k_i} = g^{x_P \cdot k_i} \cdot g^{j \cdot k_i} \cdot S_i = g^{(x_P + j)k_i} \cdot S_i = \hat{y}^{k_i} \cdot S.$$

Finally, we observe that $\log_g \hat{y} = \log_g y_P + j$, so $\log_g(y_P)$ can be determined directly from $\log_g \hat{y}$ and vice versa. $\square$

Theorem 2.3 *Let n be a natural number. Then breaking Multi-ElGamal for n addressees is as difficult as breaking ElGamal.*

Proof: Clearly, any algorithm that breaks ElGamal also breaks the Multi-version of it. So, only the implication the other way around needs to be shown. Suppose there exists an efficient algorithm $\mathcal{A}$ that on input of n sequences of h Multi-ElGamal encryptions (in the i-th encryption, $1 \leq i \leq h$, the same message S_i has been sent to all n users - with random public keys - using the same random number k_i) has a non-negligible chance of outputting (all) secret information. Now let a sequence of ElGamal encryptions for a participant P be given, say $[g, y, k_i, S_i]$ for $i = 1, 2, \ldots, h$. Then by the first part of Lemma 2.2 we can construct a sequence of outputs of a Multi-ElGamal encryption with n participants using the same k_i and S_i: the public keys of the participants will be random and the secret key of P follows from any of the secret keys of the participants. Combining this output with $\mathcal{A}$ we obtain an algorithm $\mathcal{B}$, as efficient as algorithm $\mathcal{A}$, which breaks the ElGamal encryptions for participant P with the same non-negligible chance. $\Box$

Using the ideas of [13], the ElGamal scheme can very conveniently support the construction of public keys in which the secret key is secretly shared among n share-holders (TRP's in our situation) in an n out of n secret sharing scheme. Suppose all share-holders have chosen a secret key x_i less than q and have publicized the resulting ElGamal public key $y_i = g^{x_i}$. Then, their product denoted by y, will be the shared public key. Observe that the associated secret key x is given by $\log_g y = \sum_{i=1}^{n} x_i$. The ElGamal encryption $(g^k, y^k \cdot S) = (A, B)$ of a message S with respect to the public key y, can be decrypted by a third party (a LEA in our situation) by first asking the i-th share-holder to return $A_i = A^{x_i}$ and then to calculate S by $B/\prod_{i=1}^{n} A_i$. Observe that the share-holders do not have to come together and explicitly reconstruct the secret. If, in our situation, many TRP's have publicized their public key, then users *themselves* can choose the share-holders (they trust) and form the resulting public key.

By following Pedersen [15], [16] one can, for any $1 \leq k \leq n$, construct an ElGamal public key $y = g^x$ in which the secret key x is shared in a k out of n secret sharing scheme as the constant term of a polynomial f of degree $k - 1$. Also, shareholders can verify the validity of their share. In [15] a (trusted) dealer is required to construct f. In [16] f is interactively and securely constructed by the share-holders themselves (in our situation, for instance on request of a user). As a dealer forms a single point of failure, the latter construction is preferred in our situation. As above, one can construct a protocol (also used in [5]) in which a third party (a LEA in our situation) can decrypt an ElGamal encryption $(g^k, y^k \cdot S) = (A, B)$ of a message S without the share-holders need to come together and explicitly reconstruct their secret. More precisely, consider k share-holders in the scheme with public computable $a_1, ..., a_k$ and shares $s_1, ..., s_k$ (see [15, p.223]). Then the party first asks the i-th share-holder to return $A_i = A^{s_i}$ and subsequently determines S by calculating $B/\prod_{i=1}^{m} {A_i}^{a_i}$. We note that for $k = n$, the earlier mentioned scheme is more efficient.

3 A proof of knowledge on the equality of logarithms

The following result seems to be part of the mathematical "folklore", but for the sake of completeness a proof is given in Appendix A. The result is an extension of the Chinese Remainder Theorem in the situation that not necessarily all moduli are relatively prime in pairs.

Proposition 3.1 *Let a_i, b_i for $i = 1, 2, \ldots, n$, be integers and let C_i denote the cosets $a_i + (b_i)$ in Z, where (b_i) stands for $b_i Z$. Then the following assertions are equivalent:*

1. *The intersection of all C_i's is non-empty and can be written as $y + (\mathrm{lcm}(b_1, b_2, \ldots, b_n))$ for some integer y.*
2. *Every pair of C_i's has a non-empty intersection.*
3. *$\gcd(b_i, b_j)$ divides $a_i - a_j$ for all $1 \le i \ne j \le n$.*

Now consider elements $g_1, g_2, \ldots, g_n$ (not necessarily distinct) in G. Suppose that person P (for prover) gives $h_1, h_2, \ldots, h_n \in G$ to person V (for verifier) and states:

S. There exists a number $0 < k < q$, such that for all $1 \le i \le n$

$$g_i^k = h_i, \tag{1}$$

or equivalently, there exists a number $0 < k < q$, simultaneously satisfying:

$$k \equiv \log_{g_i} h_i \pmod{\mathrm{ord}(g_i)}. \tag{2}$$

where the "ord" of a group element stands for its multiplicative order. Note that if all g_i are generators then all $\log_{g_i} h_i$ will coincide.

The following protocol lets P prove statement **S** without revealing anything about k; it is inspired by the authentication schemes of Schnorr [17] and Guillou-Quisquater [9]. Moreover, it is an extension of a signature scheme introduced by Chaum and Pedersen in [3] (an anonymous referee is thanked for this reference). In this protocol a positive integer v occurs, that will be called the *confidence* level of the protocol. We will demand that this number satisfies:

$$\begin{aligned} v \le \min\{v' \ \mid v' > 1 \text{ and, for some } i \ne j, \\ v' \text{ divides both } \mathrm{ord}(g_i) \text{ and } \mathrm{ord}(g_j)\ \}. \end{aligned} \tag{3}$$

Note that the smallest prime factor of $q = |G|$ is a lowerbound for v; equality holds if all g_i are generators of G. As a large v is desired, q should not have small prime factors.

Protocol 3.2

1. *P generates a random number l less than q, calculates $a_i = g_i^l$ for $1 \le i \le n$ and hands the a_i's over to V.*

2. *V generates a random* $0 < w \leq v$ *and presents* w *as a challenge to P.*
3. *P calculates* $z = w \cdot k + l \pmod{q}$ *and hands* z *over to V.*
4. *V verifies for all* $1 \leq i \leq n$ *that* $g_i^z = h_i^w \cdot a_i$*. If so, V will accept* **S***, otherwise he rejects it.*

We will now show that this protocol satisfies the following properties:

Completeness If statement **S** is true, then V will accept it.
Soundness If **S** is not true, then with a probability less than $1/v$ (so small) it will still be accepted by V.
Security If **S** is true, then V can not learn secret information on k by following the protocol.

The verification of the first property is straightforward. For the verification of Soundness, suppose that equality (2) does not hold, so there is no common solution to the n congruences in (2). Then, by Proposition 3.1, there exist $1 \leq i \neq j \leq n$ such that $\gcd(\mathrm{ord}(g_i), \mathrm{ord}(g_j))$ does not divide $\log_{g_i} h_i - \log_{g_j} h_j$. Let D denote the greatest common divisor of the latter two numbers, and let $v' = \gcd(\mathrm{ord}(g_i), \mathrm{ord}(g_j))/D$. Now, although P has (some) freedom in choosing $\log_{g_i} h_i$ prior to the protocol, and $\log_{g_i} a_i$ in the first step of the protocol, he has to come up with a number z in the third step satisfying for all $i,\ 1 \leq i \leq n$, and for all (or at least sufficiently many) $w,\ 0 < w < v$:

$$z \equiv w \cdot \log_{g_i} h_i + \log_{g_i} a_i \pmod{\mathrm{ord}(g_i)}.$$

The i-th and j-th congruences above (resp. modulo $\mathrm{ord}(g_i)$ and $\mathrm{ord}(g_j)$) will also hold modulo the common factor $\gcd(\mathrm{ord}(g_i), \mathrm{ord}(g_j))$, yielding:

$$w \cdot \log_{g_i} h_i + \log_{g_i} a_i \equiv w \cdot \log_{g_j} h_j + \log_{g_j} a_j \pmod{\gcd(\mathrm{ord}(g_i), \mathrm{ord}(g_j))}.$$

As $(\log_{g_i} h_i - \log_{g_j} h_j)/D$ is relatively prime with v', w is uniquely determined modulo v'. Hence the probability that V chooses the "right" w (in V's opinion) is equal to $1/v'$ which is less than or equal to $1/v$.
Finally, as an argument for Security, we assume that both P and V really choose l resp. w randomly. Observe that it is in P's best interest to do so: more uncertainty on l will give more uncertainty on k to V in the third step of the protocol. Now we will proceed with the standard zero-knowledge argument: we will show that V can generate a typical transcript $(a_1, \ldots, a_n; w; z)$ of the protocol himself, i.e. without communicating with P. To this end, V can choose w and z at random and evaluate $a_i,\ 1 \leq i \leq n$, such that they satisfy $g_i^z = h_i^w \cdot a_i$. Then it easily follows - provided P's statement is correct - that $a_i = g_i^l$ for $l = z - k \cdot w$.

Note that for Security it is required that the verifier follows the protocol, i.e. the verifier must choose his challenges w in a random way. Although intuitively clear, we can not prove that V learns no secret information by deviating from the protocol by choosing his challenges in a non-random way (cf. [3]). In the terminology of [18, Ch. 13] the above proof system for equality of logarithms is perfect zero-knowledge for an honest verifier, but we do not know whether it is perfect zero-knowledge without qualification, i.e. for any (dishonest) verifier. In our application of it in Section 4 we will enforce the verifier to be honest, i.e. to choose his challenges in a random way, thereby ensuring security.

We remark that the verification in the fourth step of the protocol can be rewritten as $g_i^z \cdot h_i^{-w} = a_i$. The use of data in the protocol can be reduced if P hands over the hash values $H_i = H(a_i)$ of the a_i - for some secure hash function $H(.)$ - instead of the a_i themselves. The verification step in the fourth step of the protocol then becomes:

$$H(g_i^z \cdot h_i^{-w}) = H_i. \tag{4}$$

A similar technique is employed in the U.S. Digital Signature Algorithm. To achieve the same level of security the number of bits in the output of the hash should not be less than $\log_2(v)$.

4 Binding the ElGamal Encryption Scheme

In this section we will present a construction for binding the ElGamal schemes using the multiuser extension discussed in Section 2. We shall do this with a (detailed) illustration, in which we will use the notation of Section 2. We will also make use of a conventional symmetric cipher $E(.)$ and of a public one-way (hash) function $H(.)$.

Suppose that Ronald from America wants to send a confidential document D to Margaret in Britain using a (government supported) Public Key Infrastructure (PKI) that incorporates binding ElGamal. Part of the PKI-policy is the choice of a confidence parameter v: the probability that binding data are accepted while the values of S sent to B and the TRP differ should be less than $1/v$. We assume that the parameters of the ElGamal system are chosen such that inequality (3) holds, that is q has no prime factors less than v. Now suppose that the national PKI-policy of America (resp. Britain) states that Ronald has to virtually address his messages to an American TRP (resp. a British TRP). Also suppose that the American PKI-policy allows the use of "splitted" public keys as explained at the end of Section 2. Let $\mathrm{TRP}_{\mathrm{A}_1}, \mathrm{TRP}_{\mathrm{A}_2}$ respectively $\mathrm{TRP}_{\mathrm{B}}$ be Trusted Recovery Parties from respectively America and Britain that Ronald trusts and chooses; $\mathrm{TRP}_{\mathrm{A}_1}, \mathrm{TRP}_{\mathrm{A}_2}$ together form $\mathrm{TRP}_{\mathrm{A}}$. Let the splitted secret keys and public keys of $\mathrm{TRP}_{\mathrm{A}_1}, \mathrm{TRP}_{\mathrm{A}_2}$ be respectively denoted by $x_{A_1}, x_{A_2}, y_{A_1}, y_{A_2}$, the shared secret key and public key (of $\mathrm{TRP}_{\mathrm{A}}$) will be denoted by $x_A (= x_{A_1} + x_{A_2})$ and $y_A (= y_{A_1} \cdot y_{A_2})$. Also, the secret key and public key of $\mathrm{TRP}_{\mathrm{B}}$ will be denoted respectively by x_B and y_B. Finally, the secret and public key of Margaret will be simply denoted by x and y.

Ronald chooses a random $k < q$ and a session key $S \in H$ and sends the following data-block to Margaret: $(E, C, R_M, R_A, R_B, bind)$ where:

C1. $E = E_S(D)$: the document encrypted by E under session key S.

C2. $(C, R_M) = (g^k, y^k \cdot S)$: the session key S enciphered with Margaret's public key;

C3. $(C, R_A) = (g^k, y_A^k \cdot S)$, $(C, R_B) = (g^k, y_B^k \cdot S)$: the session key S enciphered with the public keys of resp. $\mathrm{TRP_A}$ and $\mathrm{TRP_B}$.
C4. *bind*.

First observe that if Ronald uses the scheme correctly, then Margaret can determine S by calculating R_M/C^x; $\mathrm{TRP_B}$ can offer S to a British LEA by calculating R_B/C^{x_B}. An American LEA can ask $\mathrm{TRP_{A_1}}$ (resp. $\mathrm{TRP_{A_2}}$) to calculate $C^{x_{A_1}}$ (resp. $C^{x_{A_2}}$), and then calculate S by $R_A/(C^{x_{A_1}} \cdot C^{x_{A_2}})$. This is just an application of the multiuser ElGamal scheme which we showed to be as secure as the original ElGamal scheme.
Now we come to the construction of the binding data *bind*. Observe that the three numbers $C, R_A/R_M, R_B/R_M$ are respectively equal to $g^k, (y_A/y)^k$, and $(y_B/y)^k$, that is, they are equal to the group elements $g, y_A/y, y_B/y$ raised to the same power k. Hence, k can be viewed as the solution of the equality:

$$g^{\kappa} = C \ , \quad (y_A/y)^{\kappa} = R_A/R_M \ , \quad (y_B/y)^{\kappa} = R_B/R_M. \tag{5}$$

Now suppose we know that equality (5) has a solution k'. Given that the C and R_M are formed correctly (they are meant for Margaret to decrypt the message using ElGamal). It follows that $R_A = (y_A/y)^{k'} \cdot R_M = (y_A/y)^{k'} \cdot y^{k'} \cdot S = (y_A)^{k'} \cdot S$. That is, (C, R_A) is a well-formed ElGamal encryption of the same S for $\mathrm{TRP_A}$. A similar conclusion holds for $\mathrm{TRP_B}$.
We conclude that to construct binding data for the ElGamal scheme one only has to construct data which shows that (5) has a solution. For this one would like to use a non-interactive version of Protocol 3.2. To this end, Ronald generates a random $j < q$ and forms $bind = (D, F, I, z)$, where $D = g^j$, $F = (y_A/y)^j$, $I = (y_B/y)^j$ and $z = w \cdot k + j \pmod q$, where $w < v$ is the result of letting the one-way function $H(.)$ work - in a fixed, public way - on $E, C, R_M, R_A, R_B, D, F, I$ and possibly other public data such as Margaret's full identity and the date/time. In effect, w can not be predicted by Ronald beforehand and behaves like the random challenge in Protocol 3.2, Step 2.
Now by Protocol 3.2 anybody who has access to $R_M, R_A, R_B, bind$ and the public keys of Margaret, $\mathrm{TRP_A}$, and $\mathrm{TRP_B}$ can determine that (5) has a solution by first calculating w and then by verifying that

$$g^z = C^w \cdot D \ \ ; \ \ (y_A/y)^z = (R_A/R_M)^w \cdot F \ \ ; \ \ (y_B/y)^z = (R_B/R_M)^w \cdot I. \tag{6}$$

The probability that this verification gives the wrong answer is less than $1/v$.

As explained at the end of Section 3, one can use hashes of D, F, I in *bind* instead. The involved binding data can then be reduced to approximately the length of q. Observe that this technique can be generalized to the situation where more than two TRP's are used. For each extra TRP the binding data increases with the length of the used hash, which is rather unfortunate.

However, reducing the binding data can be done more effectively by using a standard trick of the trade (as pointed out to us by Berry Schoenmakers). Observe that from (6) it follows that one can deduce (D, F, I) if one knows (w, z).

Now we let (in the above notation) the binding data consist of (w, z) (instead of (D, F, I, z)). Verification of the binding data now consists of three steps. First one calculates (D, F, I) as indicated in (6), that is:

$$D = g^z \cdot C^{-w} \;\; ; \;\; F = (y_A/y)^z \cdot (R_A/R_M)^{-w} \;\; ; \;\; I = (y_B/y)^z \cdot (R_B/R_M)^{-w}.$$

Second (as before), let the one-way function $H(.)$ work - in a fixed, public way - on $E, C, R_M, R_A, R_B, D, F, I$ and possibly other public data such as Margaret's full identity and the date/time resulting in a $w' < q$. Third (and finally), check if w' equals w. If so accept the binding data (and conclude that (5) has a solution), otherwise reject it (and conclude that (5) has no solution). Note that one can easily convert the "new" (w, z) type of binding data to the "old" (D, F, I, z) type (and vice versa). Hence it follows that the probability that this verification gives the wrong answer is less than $1/v$.
Note that these "new" binding data are of fixed (small) length, namely the length of q plus the length of the output of $H(.)$ which is approximately equal to the length of q. Also, one can easily generalize this technique to the situation where more than two TRP's are used. The length of the binding data is independent of the number of TRP's which is very fortunate. As this technique is also more easily and securely implemented than the one using hashes of D, F, I we prefer it.

5 Conclusion

We have introduced a new concept for the establishment of an Information Security Infrastructure that does not hamper law-enforcement, using *binding data*. More in particular, we have presented a construction for binding data for the ElGamal type of public key encryption schemes using well-understood cryptographic techniques and primitives. As a side result we show that a particular simplification in a multiuser version of ElGamal does not affect its security. We expect that many more public key encryption schemes can be equipped with binding data.
A special property of the binding concept is that abuse of the system is not only difficult but also detectable by any third party (e.g. network or service provider) without harming the privacy of law-abiding users. Other properties of the binding alternative include giving users in principle a flexible choice on who to trust with their confidential communication; moreover, there need be no vulnerable parties holding (master) keys in deposit.
In our opinion, the properties of the binding alternative are flexible enough to allow cooperating countries to implement different cryptography policies on the domestic and international use of encryption in a coherent framework, which will be acceptable to many (most?) citizens in the information society. We emphasize that the binding alternative does not solve criminal encryption outside of this framework or even *within* using super-encryption - it is not meant to. Criminals can use encryption anyhow; our sole aim is that they should only be kept from effectively gaining advantage in using the (government supported) framework for this.

6 Acknowledgments

We are very grateful to Berry Schoenmakers for his valuable comments, references to existing literature and his suggestion to improve the size of the binding data at the end of Section 4.

A Proof of Proposition 3.1

We shall only show implication 2) $\Rightarrow$ 1) as the other implications are rather straightforward. To this end, we first claim that the following equality holds for all natural numbers x:

$$\gcd(x, \mathrm{lcm}(b_1, \ldots, b_n)) = \mathrm{lcm}(\gcd(x, b_1), \ldots, \gcd(x, b_n)). \tag{7}$$

This equality simply expresses that the lattice $(Z, \gcd, \mathrm{lcm})$ is distributive. For a direct verification express the integers above in terms of prime powers and use $\min\{\chi, \max\{\beta_1, \ldots, \beta_n\}\} = \max\{\min\{\chi, \beta_1\}, \ldots, \min\{\chi, \beta_n\}\}$.
The implication 2) $\Rightarrow$ 1) is trivial for $n = 2$. We shall now use induction to n. For the step $n \to n + 1$ we may assume (by the induction hypothesis) the existence of y such that $\bigcap_{i=1}^{n} C_i \ = \ y + (\mathrm{lcm}(b_1, b_2, \ldots, b_n))$. Hence:

$$\bigcap_{i=1}^{n+1} C_i = (y + (\mathrm{lcm}(b_1, \ldots, b_n)) \bigcap (a_{n+1} + (b_{n+1})). \tag{8}$$

According to the last assertion of the proposition this intersection is non-empty and of the appropriate form iff $y - a_{n+1}$ is a multiple of $\gcd(b_{n+1}, \mathrm{lcm}(b_1, \ldots, b_n))$. By equality (7) this latter equals $\mathrm{lcm}(\gcd(b_{n+1}, b_1), \ldots, \gcd(b_{n+1}, b_n))$. Hence the lefthand side of equality (8) is non-empty iff $y - a_{n+1}$ is a multiple of $\gcd(b_{n+1}, b_i)$ for $i = 1, \ldots, n$.
Now, fix i in $\{1, \ldots, n\}$ and write $y - a_{n+1} = (y - a_i) + (a_i - a_{n+1})$. Then the first term in the right hand side is a multiple of b_i and hence of $\gcd(b_{n+1}, b_i)$. The second term is a multiple of $\gcd(b_{n+1}, b_i)$ as the cosets C_i and C_{n+1} meet. So $y - a_{n+1}$ is a multiple of $\gcd(b_{n+1}, b_i)$ for each $1 \leq i \leq n$.
We conclude that the lefthand side of (8) is non-empty. That $\bigcap_{i=1}^{n+1} C_i$ is of the form $\hat{y} + (\mathrm{lcm}(b_1, b_2, \ldots, b_{n+1})$ now easily follows from the $n = 2$ case.

B An Extension for Desmedt's traceable variant of ElGamal

We use the notation of Section 2, in particular we recall that g denotes a generator of a group G. In [6], Desmedt proposes a variant of ElGamal in which all participants are given different generators by the Issuing Party (IP). Here q is a number of the form $\prod_{i=1}^{m} q_i$ where all q_i are different prime numbers. For each participant P a unique divisor $d_P \neq 1$, called P's *order*, of q is chosen (linked to P and stored). P is also given the (base-)generator $g_P = g^{q/d_P}$, the order of

which equals d_P. This generator is part of his public key of P, which also (as in the standard ElGamal) includes a $y_P \in < g_P >$ of the form $y_P = {g_P}^{x_P}$ where x_P (a random number less than q) is P's secret key. A message $S \in H$ encrypted by Ann using P's public key takes the form $({g_P}^k, {y_P}^k \cdot S)$ where k is a number less than q randomly chosen by Ann. It is shown in [6] that addressees can be identified from the (orders of the) encrypted messages sent to them. We shall refer to Desmedt's variant of ElGamal as *D-ElGamal*.
In principle, there is no need for the IP to reveal d_P to participant P. However, as can be easily seen (cf.[6]), knowledge of d_P enables the Issuing Party IP to determine S^{d_P} from the encrypted message with P's public key. So, IP can use the knowledge of the d_P to determine secret information. It can be argued (cf.[6]) that breaking the system for the IP should not be significantly easier than for an outsider. Hence, we come to the following:

Assumption B.1 *With respect to the (encryption) security of D-ElGamal we assume that the orders d_P's of participants and the factorization of q, are publicly known.*

Extending D-ElGamal to a multiuser version in a similar way as in Definition 2.1 is insecure. Indeed, suppose that a participant P wants to encrypt a message $S \in H$ meant for n participants with public keys $(g_1, y_1), \ldots, (g_n, y_n)$ in the D-ElGamal scheme; the order of i-th participant will be denoted by d_i. It seems natural, as in the conventional ElGamal scheme, that P generates one random number k and sends to the i-th participant $(g_i^k, y_i^k \cdot S)$. However, by Assumption B.1 an eavesdropper Eve can determine S^{d_i} for $i = 1, \ldots n$. So, if d is the greatest common divisor of the d_i's then Eve can also determine S^d. In other words if these d_i are relatively prime (which is likely) then Eve can determine S. Although this might be an interesting feature for some countries (sending a message to a "wrong" group of people will expose the message), it is an unacceptable security risk. Also observe that generating different k_i's for each participant doesn't help to resolve this insecurity. So, even in general, the multiuser extension of D-ElGamal is insecure.

To remedy this, we will demand in the above extension of D-ElGamal that all d_i's except for d_1 are equal to q; the resulting scheme will be called *Multi-D-ElGamal*. It should be understood that later d_1 will be used for P, the addressee. The other d_i's are for the TRP's. Of course, all k_i's are still equal to each other. Below we shall show that Multi-D-ElGamal is as secure as ElGamal with respect to g. So if the orders of all TRP's are equal to q, then session keys can be virtually addressed (as explained in the introduction) to them in a secure way. Moreover, the construction of binding data for the Multi-D-ElGamal scheme is similar to that for the Multi-ElGamal scheme, as is the splitting of private keys of TRP's. However, for reasons explained above, users should have confidence that the orders of their TRP's are in fact equal to q. A fact that is difficult to check without the factorization of q.

Let (g_P, y_P) be participant P's public key in the D-ElGamal scheme, that is $g_P = g^{q/d_P}$. For technical reasons only we introduce the *alternative D-ElGamal* *scheme*, in which the encryption of $S \in H$ takes the form $(g^k, {g_P}^k, {y_P}^k \cdot S)$, i.e. the (superfluous) element g^k is added. The *alternative Multi-D-ElGamal scheme* is formed from the Multi-D-ElGamal scheme by sending the first participant (whose order may differ from q) the alternative D-ElGamal encryption.

Lemma B.2 *If d_P is known by an attacker Ada, then breaking the alternative D-ElGamal scheme w.r.t. (g_P, y_P) is as difficult as breaking the ElGamal scheme w.r.t. g.*

Proof [sketch]: Suppose there exists an efficient algorithm $\mathcal{A}$ that after analyzing a history of encrypted messages $(g^{k_i}, {g_P}^{k_i}, {y_P}^{k_i} \cdot S_i)$, $i = 1, \ldots h$, has a non-negligible change of outputting S on input of an encrypted message $(g^k, {g_P}^k, {y_P}^k \cdot S)$.
Now suppose that participant Q has as public key y in the ElGamal scheme w.r.t. g. From this an attacker can form two public keys for two (imaginary) participants V_1 and V_2 in the D-ElGamal scheme, namely (g^d, y^d) and $(g^{q/d}, y^{q/d})$. Moreover an encryption $(A, B) = (g^k, y^k \cdot S)$ of a message $S \in H$ with Q's public key can be transformed in an encryption of $S^d \in H$ with V_1's public key, by forming (A^d, B^d). Hence, after some time, by using $\mathcal{A}$, Ada, has a non-negligible change of outputting S^d. Similarly, Ada has a non-negligible change of outputting $S^{q/d}$. As q and q/d are relatively prime (q is square-free), Ada has a non-negligible change of outputting S. □

Theorem B.3 *Let n be a natural number. Then breaking Multi-D-ElGamal for n addressees is as least as difficult as breaking ElGamal with respect to g.*

Proof [sketch]: Breaking the Multi-D-ElGamal scheme is as least as difficult as breaking the alternative Multi-D-ElGamal scheme. Now consider a sequence ("history") of h encryptions of messages S_i $(i = 1, \ldots, h)$ in the alternative D-ElGamal scheme: $(g^{k_i}, {g_P}^{k_i}, {y_P}^{k_i} \cdot S_i)$.
Observe that y_P can be seen as public key with respect to g. In fact, as $g_P = g^{q/d_P}$ and as d_P can be considered publicly known by Assumption B.1 the computation of $log_g y_P$ is as difficult as that of $log_{g_P} y_P$.
By Lemma 2.2, from a sequence of encryptions $(g^{k_i}, {y_P}^{k_i} \cdot S_i)$ anyone can construct a second sequence of encryptions of type $(g^{k_i}, \hat{y}^{k_i} \cdot S_i)$ with $\hat{y}$ random in G such that the computation of $\log_g \hat{y}$ is as difficult as that of $\log_g(y_P)$.
Anyone that chooses a random number j less than, relatively prime with q, can calculate the generator $\hat{g} = g^j$ and construct a third sequence of encryptions of type $(\hat{g}^{k_i}, \hat{y}^{k_i} \cdot S_i)$ with $\hat{g}$ a random generator in in G. It also follows that the computation of $\log_{\hat{g}} \hat{y}$ is as difficult as that of $\log_g \hat{y}$, which is as difficult as the computation of $log_{g_P} y_P$.
Hence - like in the proof of Theorem 2.3 - from the history of encryptions of messages in the alternative D-ElGamal scheme, anyone can construct a typical history of encryption of messages in the alternative Multi-D-ElGamal scheme. By a similar argument as used in Theorem 2.3, breaking the latter, means breaking the alternative D-ElGamal scheme which by Lemma B.2 and Assumption B.1 means breaking ElGamal with respect to g. □

References

1. R. Anderson, M. Roe, *The GCHQ Protocol and its Problems*, these proceedings.
2. D.M. Balenson, C.M. Ellison, S.B. Lipner, S.T. Walker (TIS Inc.), *A New Approach to Software Key Escrow Encryption*, in: L.J. Hoffman (ed.), Building in Big Brother (Springer, New York, 1996), pp. 180-207. See also http://www.tis.com.
3. D. Chaum, T.P. Pedersen, *Wallet Databases with Observers* Advances in Cryptology - CRYPTO '92 Proceedings, Springer-Verlag, 1993, pp. 89-105.
4. D. Coppersmith, *Finding a Small Root of a Univariate Modular Equation*, Advances in Cryptology - EUROCRYPT '96 Proceedings, Springer-Verlag, 1995, pp. 155-165.
5. R. Cramer, R. Gennaro, B. Schoenmakers *A Secure and Optimally Efficient Multi-Authority Election Scheme*, these proceedings.
6. Y. Desmedt, *Securing Traceability of Ciphertexts - Towards a Secure Key Escrow System*, Advances in Cryptology - EUROCRYPT '95 Proceedings, Springer-Verlag, 1995, pp. 147-157.
7. T. ElGamal, *A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory 31(4), 1985, pp. 469-472.
8. Interagency Working Group on Cryptography Policy, *Enabling Privacy, Commerce, Security and Public Safety in the Global Information Infrastructure*, 17 May 1996, see http://www.cdt.org/crypto/clipper_III.
9. L.C. Guillou, J.-J. Quisquater *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory*, Advances in Cryptology - EUROCRYPT '86 Proceedings, Springer-Verlag, 1986, pp. 123-128.
10. J. Hastad, *On Using RSA with Low Exponent in a Public Key Network*, Advances in Cryptology - CRYPTO '85 Proceedings, Springer-Verlag, 1993, pp. 403-405.
11. N. Jefferies, C. Mitchell, M. Walker, *A Proposed Architecture for Trusted Third Party Services*, Cryptography: Policy and Algorithms, Proceedings of the conference, Springer-Verlag (LNCS 1029), 1996, pp. 98-104.
12. A.K. Lenstra, P. Winkler, Y. Yacobi *A Key-Escrow System with Warrants Bounds*, Advances in Cryptology - CRYPTO '95 Proceedings, Springer-Verlag, 1995, pp. 197-207.
13. S. Micali, *Fair Public-key Cryptosystems*, Advances in Cryptology - CRYPTO '92 Proceedings, Springer-Verlag, 1993, pp. 113-138.
14. National Research Council, *Cryptography's Role in Securing the Information Society*, K.W. Dam, H.S. Lin (Editors), National Academy Press Washington, D.C. 1996, pp.720.
15. T.P. Pedersen, *Distributed Provers with Applications to Undeniable Signatures*, Advances in Cryptology - EUROCRYPT '91, Springer-Verlag, 1991, pp. 221-242.
16. T.P. Pedersen, *A Treshold Cryptosystem Without a Trusted Party*, Advances in Cryptology - EUROCRYPT '91, Springer-Verlag, 1991, pp. 522-526.
17. C.P. Schnorr, *Efficient Signature Generation for Smart Cards*, Advances in Cryptology - CRYPTO '89 Proceedings, Springer-Verlag, 1990, pp. 225-232.
18. D.R. Stinson *Cryptography: theory and practice*, CRC press, 1995, pp.434.
19. E.R. Verheul, B.-J. Koops, H.C.A. van Tilborg, *Binding Cryptography. A fraud-detectible alternative to key-escrow solutions*, Computer Law and Security Report, January-February 1997, pp. 3-14.

This article was processed using the LaTeX macro package with LLNCS style