Breaking Four Mix-related Schemes Based on Universal Re-encryption

George Danezis

K.U. Leuven, ESAT/COSIC,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium.
George.Danezis@esat.kuleuven.be

Abstract. Universal Re-encryption allows El-Gamal ciphertexts to be re-encrypted without knowledge of their corresponding public keys. This has made it an enticing building block for anonymous communications protocols. In this work we analyze four schemes related to mix networks that make use of Universal Re-encryption and find serious weaknesses in all of them. The Universal Re-encryption of signatures is open to existential forgery, and the two mix schemes can be fully compromised by an passive adversary observing a single message close to the sender. The fourth scheme, the rWonGoo anonymous channel, turns out to be less secure than the original Crowds scheme, on which it is based. Our attacks make extensive use of unintended 'services' provided by the network nodes acting as decryption and re-routing oracles. Finally, our attacks against rWonGoo demonstrate that anonymous channels are not automatically composable: using two of them in a careless manner makes the system more vulnerable to attack.

Keywords: Universal re-encryption, re-encryption mix networks, anonymous communications, traffic analysis.

1 Introduction

An important technique to achieve anonymous communication is the *mix*, an anonymizing relay, first proposed by David Chaum [1]. In his scheme messages to be anonymized, on their journey from Alice to Bob, are first encrypted under the public keys of all intermediate mixes. The messages are then relayed by all mixes in succession that decrypt them, effectively pealing off a layer of encryption at the time, and forwarding them to the next mix. As a result an observer of the network should find it hard to link senders and receivers of messages.

Many mix based systems, inspired from this architecture, have been designed and deployed [2–4]. They all use a hybrid encryption scheme, that combines the necessary public key cipher with a symmetric key cipher for bulk encryption. This technique keeps the computational cost of running a mix low, and allows more messages to be mixed together. Yet this architecture suffers from *replay* attacks: the same message, if routed twice in the mix network, will at each stage decrypt to bitwise exactly the same plaintext. To prevent adversaries from making use of this property to facilitate traffic analysis, most schemes keep track of the messages processed and refuse to process them again. The storage cost is proportional to the number of messages processed.

An alternative approach – also with the independent advantages of proofs of robustness – relies on mixed messages being re-encrypted instead of decrypted. In such schemes [5, 6] messages are encrypted using the El-Gamal public key cipher [7], and each mix node re-encrypts them on their way. Finally all messages are decrypted by a threshold decryption scheme at the end of the route. The re-encryption is randomized, and replaying a message will lead to different intermediate messages in the network. The re-encryption operation does not require any secrets, but requires the knowledge of the public key used for encryption.

Golle *et al.* [8] proposed a scheme, named Universal Re-encryption, that does away with the requirement to know the public key, under which a ciphertext was encrypted, to be able to re-encrypt it. A plaintext m encrypted under public key (g, g^x) has four components (using fresh k, k'):

$$\mathrm{URE}_x(m) := (a, b, c, d) := (g^{k'}, (g^x)^{k'}; g^k, (g^x)^k \cdot m) \tag{1}$$

Such a ciphertext can be re-encrypted by anyone, and become unlikable to the original one using fresh z, z':

$$(a', b', c', d') := (a^{z'}, b^{z'}; a^z \cdot c, b^z \cdot d) \tag{2}$$

Note that the re-encrypted product of Universal Re-encryption is a valid cipher text of message m, encrypted under the secret key x, i.e. $\mathrm{URE}_x(m)$.

The Universal Re-encryption primitive itself, and its extensions [9], are believed to be secure. In this work we study the applications of this primitive, in the context of anonymous communications, that turn out to have numerous weaknesses.

First we demonstrate that the attempt of Klonowski *et al.* [10] to make re-encryptable RSA signatures is insecure, and vulnerable to existential forgery. Then we consider the mix scheme of Klonowski *et al.* [11] and Gomulkiewicz *et al.* [12] that attempt to use Universal Re-encryption to build replay resistant mix networks. Their schemes can be attacked by a passive adversary that observes the message ciphertext at just one point, close to the sender Alice. Finally we consider the rWonGoo scheme by Lu *et al.* [13]. The scheme takes into account that the careless use of Universal Re-encryption is susceptible to tagging attacks, and a variant of re-encryption is used. Yet rWonGoo fails to protect against all attacks, and we demonstrate that it is in fact weaker then the simple Crowds [14] anonymity scheme. We propose a fix to make rWonGoo as secure as Crowds, yet the heavy cryptography used becomes superfluous.

2 Breaking the "Universal Re-Encryption of Signatures"

Klonowski *et al.* [10] extend the universal re-encryption scheme by Golle *et al.* [8], that allows ElGamal [7] ciphertexts to be re-encrypted along with a valid

RSA [15] signature. The transform is key less, and can be performed by any third party. The key feature of the Klonowski *et al.* scheme is that the signature associated with the ciphertext remains valid, despite the ciphertexts being modified through re-encryption. Schemes with such properties have the potential to be used in anonymous credential, e-cash and electronic election schemes, as well as a plethora of other application in the field of privacy enhancing technologies. Unfortunately their scheme is insecure since signed ciphertexts can be combined, without the knowledge of any signing secrets, to produce valid signatures.

Assuming that $N = pq$ with p and q being two random large primes and let g be in $\mathbb{Z}_N^*$. All operations are performed modulo N, unless otherwise stated. For a message m an authority creates an RSA signature m^d (d being its signature key). To encrypt the message to a public key $y = g^x$, the authority chooses uniformly at random two values k_1 and k_2. A cipher text in the Klonowski *et al.* scheme is composed of the following elements:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) := \\ (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; (m \cdot y^{k_0})^d, (g^{k_0})^d; (y^{k_1})^d, (g^{k_1})^d) \quad (3)$$

It corresponds to an ElGamal encryption of the message, and an ElGamal encryption of the element 1 (necessary to perform a key less re-encryption), along with an RSA [15] signature (exponentiation using d) of all these elements. To re-encrypt the ciphertext anyone can choose two values k'_0 and k'_1, an perform the following operation:

$$(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1}; \alpha_2 \cdot \alpha_3^{k'_0}, \beta_2 \cdot \beta_3^{k'_0}; \alpha_3^{k'_1}, \beta_3^{k'_1}) \quad (4)$$

Klonowski *et al.* propose to accept the signature as valid if $\alpha_0 = \alpha_2^e$ holds, where e is the public verification key, corresponding to the signature key d (the RSA property is that $e \cdot d \mod (p-1)(q-1) \equiv 1 \Rightarrow a^{e \cdot d} \mod N \equiv a \mod N$). This unfortunately does not guarantee that the ciphertext has not been modified, and does not therefore provide neither integrity nor non-repudiation as a signature scheme should.

2.1 Attacking the Scheme

The attack relies on the algebraic properties of RSA, in that the product of two signatures, results in the signature of the product, or more formally $m_0^d \cdot m_1^d = (m_0 \cdot m_1)^d$. Therefore if an attacker knows a signed plaintext $m', (m')^d$, it can construct a valid Klonowski *et al.* ciphertext by multipying it into another ciphertext in the following way:

$$(\alpha_0 \cdot m', \beta_0; \alpha_1, \beta_1; \alpha_2 \cdot (m')^d, \beta_2; \alpha_3, \beta_3) \quad (5)$$

The verification equation holds since $\alpha_0 \cdot m' = m \cdot m' \cdot y^x = (m \cdot y^k \cdot (m')^d)^e = (\alpha_2 \cdot (m')^d)^e$. The known plaintext and signature can therefore be multiplied into a valid ciphertext, at any stage, and produce another valid plaintext.

An adversary can also use two valid but unknown ciphertexts signed and encrypted to the same keys, and combine them to produce another valid, and apparently signed ciphertext.

$$(\alpha_0 \cdot \alpha'_0, \beta_0 \cdot \beta'_0; \alpha_1, \beta_1; \alpha_2 \cdot \alpha'_2, \beta_2 \cdot \beta'_2; \alpha_3, \beta_3) \tag{6}$$

Which would be a valid ciphertext since $m \cdot y_0^k \cdot m' \cdot y^{k'_0} = ((m \cdot y_0^k)^d \cdot (m' \cdot y^{k'_0})^d)^e$. Therefore an adversary can use ciphertexts, with unknown plaintexts and combine them into another valid ciphertext. This leads to existential forgery.

3 Breaking Onions Based on Universal Re-encryption

In Klonowski *et al.* [11] and Gomulkiewicz *et al.* [12] two very similar mix format schemes based on Universal Re-encryption are described. The first paper [11] discusses how such construction can be used to route messages in the mix network, including mechanisms for reply blocks and detours [4]. The second paper [12] claims that the use of Universal Re-encryption makes the mix scheme invulnerable to replay attacks. We will show that both schemes are vulnerable to tracing attacks by an adversary that observes the sender injecting an onion into the network, has the ability to use the network, and controls one corrupt mix.

The encoding schemes proposed are very simple. The sender (or a third party as described in [11]) wants to send a message m though a sequence of mixes $J_1, J_2, \ldots, J_{\lambda+1}$, to the final receiver $J_{\lambda+1}$. The public keys corresponding to each node J_i are globally known and are $y_i = g^{x_i}$. Each address in sequence and the message is universally re-encrypted using El-Gamal:

$$\mathrm{URE}_{x_1}(J_2), \mathrm{URE}_{x_1+x_2}(J_3), \ldots, \mathrm{URE}_{x_1+x_2+\ldots+x_\lambda}(J_{\lambda+1}), \mathrm{URE}_{x_1+x_2+\ldots+x_{\lambda+1}}(m) \tag{7}$$

$\mathrm{URE}_x(m)$ denotes the ciphertext one gets by performing universal re-encryption on the message m under private key x. Note that only the public component $y = g^x$ of the private key x is required to perform this operation.

Routing and decryption are taking place in parallel. The onion is first relayed to J_1, that uses its secret key x_1 to decode all blocks, retrieve J_2 and forward the message. There is no discussion in [11, 12] about removing the blocks that have been decoded, or adding blocks to pad the message to a fixed size, but these can easily be done to hide the position of different mixes on the path and the overall path length.

3.1 Attacking the Scheme

Universal re-encryption, $\mathrm{URE}_x(m)$, of a plaintext has some important properties that make our attacks possible. The ciphertext $\mathrm{URE}_x(m)$ has two components: an ElGamal encryption of 1 under the public key g^x and the encryption of the message m under the same public key.

$$\mathrm{URE}_x(m) \equiv (g_1^{k_1}, g_1^{k_1 x}, g_2^{k_2}, g_2^{k_2 x} m) \tag{8}$$

It is possible for anyone that knows $\mathrm{URE}_x(m)$ to encrypt an arbitrary message m' under the same public key. Simply chose random k_3, k_4 and encode the message m' by multiplying it by the blinded encryption of 1:

$$\mathrm{URE}_x(m') \equiv ((g_1^{k_1})^{k_3}, (g_1^{k_1 x})^{k_3}, (g_1^{k_1})^{k_4}, (g_1^{k_1 x})^{k_4} m') \tag{9}$$

Given $\mathrm{URE}_x(m)$ it is easy to further encrypt it under an additional, arbitrary, key x_a and get $\mathrm{URE}_{x+x_a}(m)$ without the knowledge of the secret x:

$$\mathrm{URE}_{x+x_a}(m) \equiv (g_1^{k_1}, (g_1^{k_1})^{x_a} \cdot g_1^{k_1 x}, g_2^{k_2}, (g_2^{k_2})^{x_a} \cdot g_2^{k_2 x} m) \tag{10}$$

An interesting property is that $\mathrm{URE}_x(m')$ is indistinguishable from $\mathrm{URE}_x(m)$ by anyone who does not know the secret key x. Even if a party knows x it is impossible to determine that $\mathrm{URE}_x(m')$ was derived from $\mathrm{URE}_x(m)$.

We further note that each mix in fact acts as a decryption oracle:

1. The mix J_i receives an onion composed of universally re-encrypted blocks.

$$\ldots, \mathrm{URE}_{x_i}(J_{i+1}), \mathrm{URE}_{x_i+x_{i+1}}(J_{i+2}), \ldots, \mathrm{URE}_{x_i+x_{i+1}+\ldots+x_{\lambda+1}}(m) \tag{11}$$

2. The Mix J_i decrypts all blocks using its secret x_i. The result is:

$$\ldots, J_{i+1}, \mathrm{URE}_{x_{i+1}}(J_{i+2}), \ldots, \mathrm{URE}_{x_{i+1}+\ldots+x_{\lambda+1}}(m) \tag{12}$$

3. The mix reads the next address J_{i+1}. If it is not well formed it stops (or starts the traitor tracing procedure described in Section 4.5 of [12]). Otherwise it re-encrypts all blocks and sends the resulting message to J_{i+1}.

Using the properties of universal re-encryption and the protocol that each mix implements an attacker that observes a message can trace it to its ultimate destination. Each block $\mathrm{URE}_{x_1+\ldots+x_i}(J_{i+1})$ is *replaced* by a block that redirects the onion to the corrupt node A followed by another block that contains the next address encrypted under the public key of the corrupt node x_a. A 'label' block that is the encryption of a fixed, per onion, label L has to also be included in oder to be able to run multiple tracing attacks in parallel.

$$\begin{aligned}&\mathrm{URE}_{x_1+\ldots+x_i}(J_{i+1})\\ &\qquad \leftarrow \mathrm{URE}_{x_1+\ldots+x_i}(A), \mathrm{URE}_{x_1+\ldots+x_i+x_a}(J_{i+1}), \mathrm{URE}_{x_1+\ldots+x_i+x_a}(L)\end{aligned} \tag{13}$$

Each mix J_i on the route will decode the message without realizing that it has been modified. Furthermore it will decode the block containing the address of the next mix J_{i+1} and the label L. The decoded message will contain:

$$\ldots, A, \mathrm{URE}_{x_a}(J_{i+1}), \mathrm{URE}_{x_a}(L), \ldots \tag{14}$$

The address A is interpreted by the honest mix J_i as the first address and the decoded message is redirected there. Once the adversary received it he can decode $\mathrm{URE}_{x_a}(J_{i+1})$ and $\mathrm{URE}_{x_a}(L)$ using his secret x_a to retrieve the next node J_{i+1} and the label L respectively.

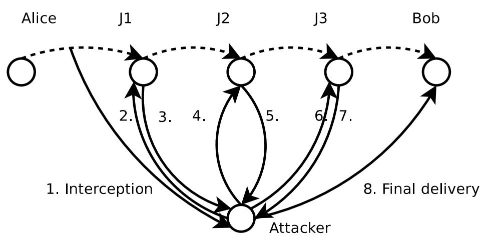

Fig. 1. After intercepting Alice's mix packet, the attacker redirects the message to themselves.

The attack results in the path of the traced onion becoming $J_1, A, J_2, A,$-$J_3, A, \ldots, A, J_{\lambda+1}$, as illustrated in Figure 1. The attacker is able to receive the onion every time it exists a mix, decode the next address and the label L, and re-insert it in the correct node to continue the tracing.

Our attack only requires a brief observation of the network to capture the onion to be traced. After that the onion is modified, and the mixes will not only decode the next address, but also forward that information to the attacker node. Therefore there is no need to perform any further passive or active attacks against messages in the network. Note that such onions can be traced even after they have been routed, since no duplicate detection mechanism is implemented. A replay prevention mechanism is difficult to implement in the context of universal re-encryption since all ciphertext (even of the same plaintext) are unlinkable without all the keys.

The fact that onions in a mix network are required to be of fixed size does not foil the attack. Since the linkage of the different parts of the message is so week, it is possible to remove the tail blocks to allow for enough space to modify the message, as described above, to trace the connection. In case the message is too short to do this, it is still possible to perform the tracing in multiple steps, that only require replacing (over-writing) one section of the message to redirect it to the adversary. Then the same message is injected in the network with the next section / header overwritten to re-direct to the attacker again until the final recipient is found.

3.2 Replay and Tagging attack

Besides the attack described above, the design in [12] fails to protect against replay attacks. An attacker can embed a tag that can be recognized after each mix J_i has processed the packet: he simply appends to or replaces the last block of the message with $\mathrm{URE}_{\sum x_i + x_a}(L)$. Once the message is processed the output will contain $\mathrm{URE}_{x_a}(L)$, which the adversary can decode to retrieve the label L. If the same message is inserted again it will output a message with the the same label, which leads to the classic replay attack.

Lu *et al.* [13] also point out that the scheme is susceptible to tagging attacks similar to those first proposed by Birgit Pfitzmann [16]. Their attack allows a corrupt receiver to trace the message and uncover Alice as its sender. They correctly point out that this attack is outside the threat model of Klonowski *et al.* [11] and Gomulkiewicz *et al.* [12], since they assume that Alice and Bob trust each other. Our attacks do not make this assumption, and allow an arbitrary third party that acts as an active adversary and controls one node to fully trace and decrypt the messages exchanged.

4 Weaknesses of the rWonGoo Scheme

Lu *et al.* [13], note that Universal Re-encryption is susceptible to tagging attacks, but also propose rWonGoo, a novel anonymous communications scheme based on re-encryption. rWonGoo was designed to protect against tagging attacks, where an adversary modifies a message to trace it through the networks, and replay attacks, where a message is replayed to help tracing. We next provide a quick description of rWonGoo that will help us highlight its vulnerabilities (a full description is provided in [13]).

rWonGoo is broadly inspired by the Crowds anonymization scheme [14], and aims to be deployed in a decentralized network of thousands of peers. It assumes that an adversary is prevented from snooping on the network by link encryption, but may also control a fraction of nodes to assist the attack. The communication in rWonGoo is divided into two phases. In the fist phase the channel is opened through the network between Alice and Bob, and the keys necessary to perform the re-encryption are distributed to all nodes through the channel. In the second phase messages between Alice and Bob can be exchanged. They start off being encrypted under the keys of all intermediary nodes, that each decrypt, re-encrypt and forward messages.

An rWonGoo channel is composed of two types of relaying nodes: those that perform re-encryption and those that are simply re-routing the message. The nodes that perform re-encryption, shall be called P_i (for $1 \leq i \leq \lambda$) with El-Gamal keys (g, y_i) respectively, while those that simply redirect shall be called Q_j (no keys are necessary since only redirection is taking place at nodes Q_j). Conceptually all communication between P nodes is done using a Crowds anonymous channel over Q nodes. In some sense rWonGoo routes already on top of a crowds anonymous channel. The final node P_λ is assumed to be Bob, the ultimate recipient of the anonymous messages from Alice (also P_0).

The channel establishment protocol is of special interest to an attacker. Alice first picks a node P_1 and extends her tunnel to it. This extension is done using the crowds protocol, until node P_1 is reached. The node P_1 sends back to Alice a set of potential next nodes, with their IP addresses, TCP ports and El-Gamal public keys. Alice chooses one of them and, through an encrypted channel described below, extends her tunnel to P_2. The communications between P_1 and P_2, are using the crowds protocol. This is repeated $\lambda - 1$ time until Alice instructs $P_{\lambda-1}$ to connect to Bob.

All communications between Alice and node P_i (including Bob i.e. P_λ) are encrypted in a layered manner. Alice always knows the public keys $y_1 \ldots y_i$ and uses them to generate a *key distribution* message that distributes to all intermediates $P_1 \ldots P_i$ the keys necessary to re-encrypt messages. These are conceptually the composite public keys under which the messages seen by each P_i are encrypted. Alice sends the key distribution message:

$$A \to P_1 : ((y_1 \cdot \ldots \cdot y_i)^r, g^r; y_0^{r'}, g^{r'}) \equiv (P_0^{\text{forward}}, P_0^{\text{backward}}) \tag{15}$$

P_1 removes his key from the first part of the message, to retrieve the public key $P_1^{\text{forward}} := ((y_1 \cdot \ldots \cdot y_i)^r / (g^r)^{x_1}, g^r) \equiv (y_{1f}, g_{1f})$ necessary to re-encrypt messages traveling forward in the channel. Similarly he adds his public key to the second part of the message to calculate the key $P_1^{\text{backward}} := (y_0^{r'} \cdot (g^{r'})^{x_1}, g^{r'}) \equiv (y_{1b}, g_{1b})$ necessary to re-encrypt messages traveling back towards Alice. P_1 then sends the new key set $(P_1^{\text{forward}}, P_1^{\text{backward}})$ to node P_2. This procedure is repeated by all P in the channel, until the final message arrives at P_i:

$$P_{i-1} \to P_i : (y_i^r, g^r; (y_0 \cdot \ldots \cdot y_{i-1})^{r'}, g^{r'}) \tag{16}$$

This key distribution procedure ensures that all intermediate P_i know the public keys under which the messages they receive on the forward and backward path are encrypted. As a result they can decrypt them and re-encrypt them on their way. Upon receiving a message $M := (a, b)$ node P_j performs the decryption using its secret key (g, y_j, x_j) and re-encryption using the key $(g_{j(b|f)}, y_{j(b|f)})$, under which the message is encrypted, and passes the resulting M' to the next node in the path (using Crowds as transport).

$$M' := \text{ReEnc}_{(g_{j(b|f)}, y_{j(b|f)})}(\text{Dec}_{(g, y_j, x_j)}(M)) \tag{17}$$

Following this process a message sent from Alice to Bob encrypted under key P_0^{forward}, arrives encrypted under Bob's key (g, y_λ), and a message send back from Bob to Alice under key $P_\lambda^{\text{backwards}}$ arrives encrypted under her key (g, y_0).

4.1 Attacking rWonGoo: Capturing the Route

The key vulnerability of rWonGoo is that it is susceptible to man-in-the-middle attacks, that allow the rest of the channel to get captured after a malicious node is encountered. This means that after Alice chooses a bad node to include on the channel path, all subsequent nodes can be made to be bad too. The intuition behind this attack is that Alice knows very little about the network, and relies on intermediaries to discover other nodes and their public keys. She is therefore unable to tell the difference between a genuine interaction with the network, and a interaction that is simply simulated by an adversary.

The attacks proceeds quite simply: we assume that there is a first dishonest re-encrypting node on the path, named P_m. Once the dishonest node P_m receives the request to extend the channel, it starts simulating a network of nodes P_{m_k}, and provides Alice with their fictitious IP addresses, TCP ports and public keys (for which P_m knows the secret component). Alice chooses one of them to extend her tunnel, but no matter which one she chooses P_m never forwards any message but keeps simulating more nodes, all running the rWonGoo protocol with Alice. Finally Alice connects to Bob, directly through P_m. Note that the fact that Alice is provided a choice of nodes to chose from does not eliminate any attacks, since they are all corrupt, or even non-existent. As Alice does not have any first hand experience of any of the nodes she is asked to choose (she cannot even query them to see if they exist, since this would reveal she is the originator of the tunnel), the attacker can populate these choices with not only malicious but also fictitious nodes.

During the key distribution phase of the protocol the malicious nodes substitute the keys communicated to Bob, for use in the backward channel, with their own keys. Therefore the key distribution message received by Bob is $(y_\lambda^r, g^r; y_m^{r'}, g^{r'})$, where y_m is the public key of the adversary. As a result *any message sent by Bob back to Alice can be read by the malicious nodes.* Those messages can then be re-encrypted under the key f_m^b and sent to Alice.

Our attacks so far allows an adversary to perform a predecessor attack [17], and probabilistically find Alice after she engages in consecutive interactions with Bob. We can estimate how long, in terms of the number of fresh channels Alice has to open to Bob, the attack is likely to take. We assume that a fraction f of the network is controlled by the adversary [18]. The intersection attack succeeds immediately (for reasons explained below) if the first Crowds node after Alice, Q_1 belongs to the adversary, which it is with probability f. Consider the random variable L, which denoted the number of fresh rWonGoo channels that Alice opens to Bob, until a channel in which the first node Q_1 is corrupt. The random variable L follows a geometric distribution with parameter f, and Alice is on average expected to have $\mathbb{E}(L) = (1-f)/f$ secure anonymous tunnels until her association with Bob is uncovered.

4.2 Decrypting any Message Using Re-routing Oracles

First we note that any node in the network, including Alice and Bob, can be used as a *decryption oracle* for messages encrypted under their keys. During the key setup operation a node is asked to effectively decrypt the first part of the message it receives and relays it to the next node on the path. Consider the victim node P_i with public key y_i which is to be used to decrypt a ciphertext $m := (a, b) \equiv (g^k, y_i^k m')$. The adversary sets up an rWonGoo channel P_m, P_i, P'_m, where the nodes P_m and P'_m are controlled by the adversary. Then P_m sends to P_i the following message, that is to P_i indistinguishable from a key distribution message (k is a random factor chosen by the adversary):

$$P_m \to P_i : (b \cdot k, a; y_m^{r'}, g^{r'}) \tag{18}$$

The node P_i removes its key from the first component of the message and sends the result to the next node P'_m, which is also controlled by the adversary. The new message will be:

$$P_i \to P'_m : (b \cdot k / a^{x_i}, a; \ldots) \tag{19}$$

As a result P'_m gets $b \cdot k / a^{x_i} = y_i^k m' \cdot k / y_i^k = m' \cdot k$ and can divide it by the known factor k to retrieve the encrypted message m'. We will denote the decryption of a ciphertext m by the adversary as $m' = \mathrm{Dec}_i(m)$, which only takes subscript i (and not the private key x_i) since it can be performed even if just the name of the node is known[1].

We have shown in the previous section that a malicious P_m can always uncover the receiver P_λ (or Bob) of any message seen, and see in clear all messages send by Bob to Alice. Since any malicious node can also force any other node in the network to act as a decryption oracle, it follows that the attacker can also see in clear all messages sent by Alice to Bob. Each ciphertext m destined to Bob, has to travel through P_m, and is encrypted only under Bob's public key. The attacker can just use Bob as an oracle to retrieve the plaintext $m' = \mathrm{Dec}_\lambda(m)$.

4.3 Using any Q_m to Attack the Crowds Routing

In rWonGoo communication between any two P is done using the Crowds protocol, and we name the nodes that merely perform crowds redirection Q_i. Those simply forward the message and perform link encryption.

First, using the decryption attacks presented above, any corrupt Q_m node can capture the rest of the route until Alice asks to be connected to Bob. This is possible because the corrupt Q_m sees all the key distribution and actual messages that are relayed, starting from the first message in which Alice asks to have the rWonGoo channel connected to the next P_i on the route. At this point the corrupt rerouting node Q_m uses P_i as a decryption oracle to retrieve all information sent by Alice. As a result Q_m can simulate all interactions where the secret keys of P_i are needed, without ever relaying the channel through it. Our route capture attacks can now be performed by *any* corrupt P_m or just Q_m node on the path.

Secondly we note that a Q_m can test whether its predecessor is Alice by using it as a decryption oracle on a backwards message (which is only encrypted under Alice's key), and checking if the result is plaintext. In case the result is plaintext, Q_m can confirm that its predecessor is Alice. This turns the predecessor attack into an exact attack, and makes rWonGoo weaker than the original Crowds. Similarly the attacker can test any other node in the network to see if it is the originator of the message. This breaks anonymity after at most $\mathcal{O}(N)$ decryptions, where N is the size of the network, by a Q_m between Alice and P_1.

A confirmation attack can be mounted by any Q_m, even if it is not on the Crowds route of the first hop between P_0, or Alice, and the first mix P_1. Any Q_m observes in clear the key $((y_0 \ldots y_i)^{r'}, g^{r'}) \equiv m_s$, which is the combination

[1] Note that if the message decrypted using P_i as an oracle *is not* encrypted under the corresponding key y_i, it will result in a plaintext that is indistinguishable from random. This property can be used to detect valid decryptions, when the correct plaintext is expected to have some structure. In case the correct plaintext is also indistinguishable from random for the adversary, it is difficult to tell if the correct or incorrect node was used as a decryption oracle.

of all the public keys of the P_i's so far on the route. Q_m wants to test whether the path used is made of the guess set of nodes $P_{j_0} \dots P_{j_k}$. To do this Q_m can consecutively decrypt, using each of the nodes $P_{j_0} \dots P_{j_k}$ as oracles message m_s, (i.e. $m'_s = \text{Dec}_{j_0}(\dots \text{Dec}_{j_k}(m_s)))$. If it is the case that the plaintext equals one ($m'_s = 1$), then the guess is correct, and Q_m has established that the path so far was made of the nodes in the guess set. This is an all-or-nothing test that provides no partial information. As a result it does not scale well with the number of honest network nodes N and the path length l, since Q_m will have to perform $c = \binom{N}{l} \cdot l$ decryption requests.

4.4 The Stronger Crowds the Weaker rWonGoo

The complexity of the attack presented above, in terms of the parameters of the system, is counter intuitive. The attack becomes more difficult as the number of honest Ps that re-encrypt the messages increases before the message is 'seen' by either a dishonest P_m or a dishonest Q_m (a node that only performs Crowds between P nodes, yet can see the ciphertext and perform the guessing attack). In case the message is seen by a corrupt Q_m as it is traveling between Alice and P_1, only $\mathcal{O}(N)$ decryptions are required.

We assume that until a corrupt P_m or Q_m is reached, say node number v (at which point we can capture the route or perform the guessing attack) all nodes are selected uniformly at random. This allows us to calculate the expected position v of the first corrupt node, if we know that a certain fraction f of the network is corrupt. The number v follows a geometric distribution with parameter f and its expected value is $\mathbb{E}(v) = (1 - f)/f$. As the fraction of corrupt nodes increases we expect the message to be seen by the attacker earlier.

At the same time the Crowds protocol can be tuned with a parameter h, which is the probability a message is forwarded to its final destination (versus being forwarded to a random member of the crowd) by each node that receives it. It is also trivial to see that the average length u of each journey into the crowds subsystem (that is used to route between Ps) follows a geometric distribution with parameter h, with average path length $\mathbb{E}[u] = (1 - h)/h$.

As mentioned before our guessing attack is most effective when the number of Ps on the route is small, before the message is seen by the adversary. We know that on average the message will be seen in $\mathbb{E}(v) = (1 - f)/f$ hops, but the average length of its first Crowds trip between Alice (P_0) and the first re-encryptor P_1 is expected to be $\mathbb{E}[u] = (1 - h)/h$. We can conclude that if the parameter h is smaller than f (the corrupt fraction of nodes in the network) it is expected on average that the attacker will see the message on its first hop and be able to perform the most trivial guessing attack. The adversary only has to perform at most N decryption operations until Alice is revealed.

This result is counter-intuitive: the parameter h being smaller means that the number of intermediaries in the Crowds protocol is larger. One should expect this to increase the anonymity of the system. Contrary to this, increasing the length of the crowds path allows the adversary to observe the raw message earlier with higher probability, despite link encryption. Since the rWonGoo scheme is very vulnerable when the attacker can observe messages early on, increasing the 'anonymity' provided by the Crowds transport, decreases the overall anonymity of the system.

4.5 Partial fix for rWonGoo

As it stands rWonGoo is weaker than Crowds (that only uses link encryption, and no other cryptographic protection.) It is possible to make its security as good as Crowds with a minor modification: the sender Alice, chooses a fresh key pair (g, y_0) for each channel. This would defeat the confirmation attacks that make rWonGoo weaker than crowds, since Alice cannot be forced to decrypt a ciphertext correctly, confirming that she is the sender. This makes rWonGoo as strong as Crowds, but much more complex and unnecessarily costly at the same time.

5 Conclusion

The properties of RSA that make the 'Universal Re-encryption of signatures' scheme vulnerable to our attacks have been known, and used in the past by Birgit Pfitzman to break anonymous communications schemes [16]. To overcome them special padding schemes such as PKCS#1 [19] are used to give ciphertexts a special structure that is infeasible to reconstruct by multiplying different ciphertexts together. These padding schemes require a verifier to have access to the message plaintext in order to verify its validity, making it therefore impossible to check the validity of re-encrypted ciphertexts (since they still hide the message m). To allow decrypted ciphertexts to be verified using a signature scheme none of the fancy cryptography is necessary: it is sufficient to encrypt using Golle *et al*, a signed message, and transmit the corresponding ciphertext. The receiver then decrypts the ciphertext and can check the signature. Therefore we see little hope in fixing this scheme while retaining its interesting re-encryption properties.

The attack against the onions based on universal re-encryption is possible because of many factors: We can modify the onions, since their integrity is not protected, and their different parts are not linked to each other in a robust manner. The mixes allow themselves not only to be used as decryption oracles for arbitrary ciphertexts, but also can be used to redirect traffic to the attacker making tracing effortless. Our attack shows that the claim in section 4.3 of [12], that the insertion of blocks in the onion structure is not possible, is unfounded which directly leads to our attack.

Finally we show that rWonGoo is a very fragile scheme. The additional cryptography in rWonGoo has made the overall system more susceptible to attack, than the original Crowds proposal, that only used link encryption. In particular it is possible for all messages between Alice and Bob to be read by the adversary with high probability, following route capture. Since any participant acts as a decryption oracle, it is possible to mount confirmation attacks to find Alice more quickly than if simple Crowds was used.

Our attacks lead to two important and novel intuitions, that anonymous communication system designers should carefully take into account in the future. First, the weakness of the rWonGoo scheme demonstrates that anonymous channels are not automatically composable: rWonGoo using the crowds protocols as a transport between mixes makes the system more vulnerable, not stronger. Furthermore choosing more secure parameters for the Crowds transport used in rWonGoo, makes the overall scheme less secure, which is highly counter-intuitive.

Second, our attacks against the mix and signature schemes based on Universal Re-encryption, demonstrate the inherent difficulty in using this primitive in a secure fashion. Its power comes from its neat structure, which allows for re-encryption given only a ciphertext, and the use of multiple keys along with incremental decoding. It is these properties that made it a promising primitive for anonymous communications.

On the other hand to preserve these properties, and allow ciphertext to be universally re-encryptable, a designer is forced to let them be malleable, leak the public keys used, and is unable to add any redundancy for integrity checking of messages on their way. That is the weakness our attacks exploited, and it is a weakness that should have been foreseen given the rich literature on attacking re-encryption networks. The literature on (non-Universally) re-encryption networks demonstrates that, to be secured, such schemes require, identification of senders, expensive zero-knowledge proofs of knowledge of the plaintexts, and proofs of correct shuffle and threshold decryption. Such proofs have not yet been adapted to Universal Re-encryption, and would be difficult to adapt them to the dynamic setting of free-route mix networks, and the multiple threats that such networks face (dynamic membership, sybil attacks,. . .). Unless there is a breakthrough in this field, Universal Re-encryption will should always be used, in this context, with uttermost care.

Acknowledgements

Ross Anderson, Philippe Golle, Ari Juels have provided feedback that has improved this work. Discussions, over they years, with Paul Syverson about how to build mix networks based on Universal Re-encryption were invaluable to understand how to attack them. George Danezis is supported by the Cambridge-MIT Institute (CMI) project on 'Third generation peer-to-peer networks', the Flemish Research Council (FWO), and by the European Union PRIME project.

References

1. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
2. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003.
3. U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. Technical report, Network Working Group, May 25 2004. Internet-Draft.

4. Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium — NDSS '96*, pages 2–16, San Diego, California, February 1996. IEEE.
5. Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In Tor Helleseth, editor, *Advances in Cryptology (Eurocrypt '93)*, volume 765 of *LNCS*, pages 248–259, Lofthus, Norway, 23-27 May 1993. Springer-Verlag.
6. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Pierangela Samarati, editor, *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 116–125. ACM Press, November 2001.
7. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.
8. Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's track*, San Francisco, USA, February 2004.
9. Peter Fairbrother. An improved construction for universal re-encryption. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 79–87. Springer, 2004.
10. Marek Klonowski, Miroslaw Kutylowski, Anna Lauks, and Filip Zagorski. Universal re-encryption of signatures and controlling anonymous information flow. In *WARTACRYPT '04 Conference on Cryptology*, Bedlewo/Poznan, July 1–3 2004.
11. Marek Klonowski, Miroslaw Kutylowski, and Filip Zagrski. Anonymous communication with on-line and off-line onion encoding. In Peter Vojts, Mria Bielikov, Bernadette Charron-Bost, and Ondrej Skora, editors, *SOFSEM 2005: Theory and Practice of Computer Science, 31st Conference on Current Trends in Theory and Practice of Computer Science*, Lecture Notes in Computer Science, pages 229–238, Liptovsk Jn, Slovakia, January 22–28 2005,. 3381.
12. Marcin Gomulkiewicz, Marek Klonowski, and Miroslaw Kutylowski. Onions based on universal re-encryption – anonymous communication immune against repetitive attack. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications, 5th International Workshop, WISA 2004*, volume 3325 of *Lecture Notes in Computer Science*, pages 400–410, Jeju Island, Korea, August 23–25 2004. Springer.
13. Tianbo Lu, Binxing Fang, Yuzhong Sun, and Li Guo. Some remarks on universal re-encryption and a novel practical anonymous tunnel. In Xicheng Lu and Wei Zhao, editors, *ICCNMC*, volume 3619 of *Lecture Notes in Computer Science*, pages 853–862. Springer, 2005.
14. Michael K. Reiter and Aviel D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–38, 1999.
15. Ron L. Rivest, A. Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
16. Birgit Pfitzmann. Breaking efficient anonymous channel. In Alfredo De Santis, editor, *Advances in Cryptology (Eurocrypt '94)*, volume 950 of *LNCS*, pages 332–340, Perugia, Italy, 9-12 May 1994. Springer-Verlag.
17. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 7(4):489–522, 2004.

18. John R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.
19. PKCS #1 v2.1: RSA Cryptography Standard. RSA Security Inc., June 2002.