Kleptography trapdoor free cryptographic protocols

Bohdan Kovalenko[1] and Anton Kudin[2]

[1] National technical University of Ukraine "KPI", Kyiv, Ukraine,
animantbk@gmail.com
[2] National Bank of Ukraine,
pplayshner@gmail.com

Abstract. *Context.* Methods of known kleptography implementations are being investigated. The article focuses mostly on SETUP design of subliminal data leakage channels.
Aim. Suggest approach to develop SETUP resistant cryptosystems.
Methods. The necessary conditions for SETUP implementation is building in entropy source (otherwise generated secret will be predictable). In this article it's considered subscriber whose protocol implementation is suspected to be modified by Developer (malicious actor who is able to influence on cryptosystem implementation) to create subliminal leakage channel. Possible countermeasure is to prohibit usage own random sources for subscribers, enforce generate random values from public counters.
Results. Formal model for basic SETUP scheme has been suggested. Approach to develop of SETUP resistant protocols has been described. Two basic SETUP-resistance protocols (nonce generation protocol and Diffie-Hellman key agreement protocol) have been proposed.

Keywords: kleptography, SETUP, subliminal channel, secret leakage, DLP

Introduction

The "kleptography" term was proposed by Adam Young and Moti Yung in 1996 [YY96]. Kleptography covers methods of implementation so-called "back-doors" or "subliminal channels" in cryptosystems that allow Developers to access certain secret information from customer's cryptosystem. Developer is a kind of malicious actor who is able to make changes (illegally) into some parts of cryptosystem implementations. In general, by the level of construction, it is possible to distinguish two classes of kleptography mechanisms: embedded at the cryptosystem design stage or back-doors injection into the existing cryptosystem or protocol. Mechanisms of the second type, namely, SETUP (Secretly Embedded Trapdoor with Universal Protection) schemes, were proposed by A. Young and M. Yung [YY96]. The main idea of this scheme is that there is an additional protocol role – "Developer". Developer modifies the protocol/cryptosystem implementation on one of the endpoints (through unauthorized access to endpoints or via infection with malicious software) in such way that the victim begins to transfer a certain secret to the Developer, moreover other subscribers are unable to detect it.

The objectives of kleptography in the context of cryptographic protocols are:

1. Design of back-door in standard protocols
2. Back-door disclosure
3. Design of cryptographic protocols which are kleptography back-door free

Many examples of theoretical and practical schemes for constructing channels of imperceptible secret leakage in cryptographic schemes are known. In this paper, the main attention is paid to solving the problem of determining the conditions that provides resistance against some back-door injection attacks.

1 Kleptography mechanism classification

There are problems in the classification of kleptography systems now. It is due to the fact that there are no clear formal model of kleptography mechanisms. Kleptography covers a wide range of attacks on cryptomechanisms and attacks based on cryptomechanisms. In particular, it includes methods of constructing channels with secret leakage in standard cryptosystems (including steganographic methods), design of built-in leakage channel, development of cryptoprimitives with partial violation of cryptographic properties, etc. If we consider the methods of leakage channels constructing, we may focus on asymmetric cryptography to deliver secret to Developer or we can suspect a steganography channel that uses a system randomizer (for example, the salt of the hash function or the initial vector of the stream cipher). Cryptoprimitive with partial breach cryptographic properties can be modified to allow Developers (and no more) to decrypt messages without the knowledge of the key, or it may be a hash function that allows the designer to perform efficient preimage search for given digest or to generate 1-st or 2-nd type of collisions.

The classification complexity follows by the fact that some kleptosystems can be detected on because it's design looks to be a "nonstandard" scheme, but this can also be explained by design fault. For example, until now, it's impossible to say exactly whether the DES encryption algorithm is kleptography cipher or not, since nobody really knowns were the weak security parameters applied meaningly or this is a result of inadequate analysis. In order to create classification we consider model of kleptography system which consist of such parts: secure channel for the information transmission, users, attacker, and Developer who has a subliminal leakage channel from some of users. In this case, the attacker should not be able to violate the security of the main transmission channel. Neither the attacker nor the user should be able to violate the security of the secret leakage channel. So, we can propose such a classification of kleptography systems (depending on different criteria of classification).

By implementation availability:

1. Open standard implementations (software libraries, circuit design, RFCs).
2. Proprietary, mostly hardware implementation (cryptoprocessors, hardware cryptomodules).

By destruction level:

1. With the possibility of further usage (reverse engineering of program components, delayering of masked hardware components, specifications analysis, etc.).
2. Without further usage (probing of cryptographic controllers, built-in EEPROM memory, etc.).

By design method:

1. Modification of existing cryptosystems (insertion of a leakage channel).
2. Design of new crypto algorithms with built-in back-door.

By implementation way:

1. Modification of working cryptosystems on the user's side (through malicious software, cyber attacks, etc.).

2. Open distribution of implementations with kleptography modifications (e.g., in opensource or freeware software components).

3. Distribution of proprietary protected cryptosystems such as hardware modules.

4. Lobbying for the cryptography standardization, imposing their use through legal mechanisms or corporate policies.

According to this classification we can determine the main kleptography problems:

1. Implementation of information leakage channels through modification of standard cryptosystems implementations

2. Development of cryptosystems with built-in leakage channel

3. Search for leakage channels in a cryptosystem

4. Search for leakage channels in the implementation of the cryptosystem

5. Designing of cryptosystems without leakage channel

6. Implementation of cryptosystems that are resistant against kleptography modification

We suggest to discuss partial cases of designing of back-door free cryptosystems.

2 SETUP mechanism

2.1 Definition

One of the methods to build kleptosystems based on cryptosystem is SETUP [AOS12](Secretly Embedded Trapdoor with Universal Protection).

The main idea of SETUP is that the attacker (Developer) modifies the implementation of the standard cryptosystem in such a way that the cryptographic properties of the system are not fulfilled for the Developer but for other users it remains the same security level. Moreover, other participants can not even suspect the fact of such a modification. In addition to the SETUP also one can distinguish weak and strong SETUP.

SETUP cryptosystem C is called its modification C' that:

1. Interface interaction (input and output parameters) with C 'corresponds to the one declared for the C standard.

2. C' is efficiently computed.

3. The Developer's secret is available only to him and not contained in C'.

4. The secret information that C' sends via back-door can be decrypted effectively only by the Developer (the Developer uses his secret key for decryption).

5. No one, except the Developer, can distinguish outputs of systems C 'and C in polynomial time.

6. After analysis of the modified implementation (obtaining all necessary algorithms, destructive reverse engineering) it is impossible to restore the previous keys or to predict the future ones.

Weak SETUP is called SETUP for which both Developer and implementation owner are able to distinguish outputs C and C'. A strong SETUP is called SETUP with an additional condition - it is impossible to restore the previous keys and to predict the next keys after online analysis (non-destructive) of working system. An important parameter of the SETUP mechanism is a bandwidth. (n, m)-scheme leakage is called the SETUP mechanism, which requires to transmit m messages to a communication channel to perform leakage of n messages.

Currently, all well-known SETUP mechanisms are based on asymmetric cryptography for many cryptosystems: DSA digital signature ([AOS12]), systems based on discrete logarithm problems ([YY97]) and RSA ([YY96]).

2.2 The Discrete Logarithm Problem based protocol

In 1994, A. Young and M. Yung suggested a method for SETUP building for protocols whose security is based on DLP. The idea is that the subscriber who generates key pairs and publishes public one is modified by Developer to enable hidden data transmission to the Developer side.

Input: g – generator of multiplicative group F_p^*, $(x \in F_p^*, Y = g^x)$ – Developer's key pair,
$W, a, b \in F_p^*$ – fixed parameters

1 Generate random key $c_1 \in F_p^*$. c_1 is stored for next time.
2 Compute and publish the first public key $\underline{M_1 = g^{c_1} \mod p}$
3 Generate random $t \in \{0, 1\}$
4 Compute $z = g^{c_1 - Wt} Y^{-ac_1 - b} \mod p$
5 Compute next secret key: $c_2 = hash(z)$, $hash : \{0,1\}^* \to F_p^*$
6 Compute and publish the second public key $\underline{M_2 = g^{c_2} \mod p}$

Algorithm 1: SETUP in Diffie-Hellman protocol: transmission private key to Developer

The Developer restores the secret key c_2 using intercepted data from the open channel:

1 $r = M_1^a g^b \mod p$
2 $z_1 = M_1 / r^x \mod p$
3 $c_2 = hash(z_1)$ or $c_2 = hash(z_1 / g^W \mod p)$

Algorithm 2: SETUP in Diffie-Hellman protocol: restoring key by Developer

Thus, the Developer obtains a secret key c_2 and nobody is able to obtain it without the knowledge of the Developer's secret key x.

2.3 RSA example

In the paper [YY96] there is scheme of a leakage channel implementation into the implementation of the RSA algorithm. The main idea is that the victim's implementation generates the RSA parameters in such a way that the attacker, knowing the secret of the mechanism, is able to effectively solve the factorization problem.

Generation of original RSA parameters includes the generation of two large primes $p, q : \#p = \#q = m, m > 1024$ ($\#(\cdot)$ – bit length), these values are the secret parameters. Public parameter is $n = p \cdot q$. If an attacker modifies the cryptosystem implementation so that it can effectively factorize n, he also be able to decrypt any encrypted data. The scheme proposed in [YY96] based on the trick when RSA secret key is generated using

Developer's public key $y = g^x mod P$, where P, y, g – public parameters, $\#P = \#p = m$, $g, x, y \in F_P^*$. The parameters $W, t, a, b \in F_P^*, hash(\cdot)$ means the same as for leakage channel in the Diffie-Hellman protocol, $G : F_P^* \times K \to \{0,1\}^m$ – symmetric encryption K – key space. Also we need L – small integer and fixed symmetric key $k_0 \in K$.

1. Generate randomly $c_1, t : c_1 < P - 1, t \in \{0,1\}$.
2. Compute z as solution of the equation $y^{ac_1+b} g^{Wt} z = g^{c_1} mod P$. In order to make a value z was with a uniform distribution among m-bit values, goto step 1 until z will be in $2..2^m - 1$
3. $z' = hash(z)$
4. The least significant bit of z' set to '1' (z' should be odd).
5. Generate a secret parameter (large prime) p that has structure $p = z' + num$ (num is the smallest positive integer so that p is prime).
6. For $0 \le i \le L$:
   (a) Compute $U = G(g^{c_1}, k_0 + i)$.
   (b) Generate random $R \in \{0,1\}^m$.
   (c) Solve q and r from the equation $[U|R] = pq + r$ ($[\cdot|\cdot]$ – concatenation of bits, $r < p$). Goto step 6b until q will be prime.
   (d) Compute $n = pq = [U|R] - r$
   (e) Compute RSA exponents e, d.

Attacker can restore p and q based on secret x as follow:

1. Compute U with n (most significant bits m).
2. For $0 \le i \le L$:
   (a) Compute $m = G^{-1}(U, k_0 + i)$.
   (b) Compute $z : m^{xa} g^{b+Wt} z = m(\mod P)$.
   (c) $z' = hash(z)$
   (d) The least significant bit z' set to '1' (z' should be odd).
   (e) Compute minimal s, $p = z' + s$ – prime.
   (f) If $p|n$, attacker factorizes n and gets access to the leakage channel.

If attacker has no access to secret x, the problem of computation $z = g^{c_1 - Wt} y^{-ac_1 - b}$ (and p) will be reduced to the DLP.

3 SETUP-resistant cryptosystems

3.1 Complexity model

To assess the security of the kleptography system, it is first necessary to determine the complexity model. In cryptography, there are several basic models: computational complexity theory, Shannon's informational theory, reduction to ideal primitives etc. These models are not always adequate for kleptography mechanism. For example, if the complexity of back-door insertion in a symmetric cipher is exponential from internal state size, but nonetheless, required time for building in is practical, then such a back-door

deploying method is acceptable for us. On the other hand, if the asymptotic complexity the back-door detection is a high degree polynomial, but for the given pre conditions this time is impractical, then we may consider the back-door to be secure in practical sense.

Come Berbain and Henri Gilbert [BG07] describe practice algorithm execution time based complexity. In current article we suggest to apply a similar model, that is based on practical time complexity of real algorithm execution.

Definition 1. (Practical ensemble distinguisher) Let's consider two ensembles $E = \{e_1, e_2, ...\}$ and $E' = \{e'_1, e'_2, ...\}$, $e_i, e'_i \in S$, S is a finite set. Also, there is additional time threshold t – it's a maximal time period for algorithm execution (e.g., $t = 2^{80}$).

Practical ensemble distinguisher is probabilistic algorithm A_t which is bounded by execution time t for vector size l, $\vec{\nu} \in E^l \cup E'^l$, that returns:

$$\begin{cases} A_t(E, \vec{\nu}) = 1 & \Leftrightarrow \vec{\nu} \in E^l \\ A_t(E', \vec{\nu}) = 1 & \Leftrightarrow \vec{\nu} \in E'^l \end{cases}$$

Let's define advantage for a practical distinguisher in ensemble recognition as $Adv_{A_t}(E, E', l) = |P\{A_t(E, \vec{\nu}) = 1\} - P\{A_t(E', \vec{\nu}) = 1\}|$, where $\vec{\nu} \in E^l \cup E'^l$ - random vector with size l.

Definition 2. (Practical indistinguishability) Let's call pair of ensembles E and E' practical indistinguishable if for given security level t, maximal recognition advantage for all practical algorithms will be negligible for this security level: $Adv(E, E') = \max_{l, A_t}\{Adv_{A_t}(E, E', l)\} < \varepsilon(t)$, where $\varepsilon(t)$ – threshold value for "negligible" probability (e.g., $\varepsilon(t) = 2^{-40}$ when $t = 2^{80}$).

Further, practical indistinguishability for E and E' will be denoted as $E \simeq_t E'$.

3.2 SETUP: formal model

SETUP mechanism was suggested by M.Yung and A.Young [YY97], it isn't strictly formalized, that complicates security evaluation process. One of basic crypto systems is "challenge-response" protocol scheme, that may describe almost all single- and two-pass protocols, e.g. cipher based authentication, Diffie-Hellman key agreement. So, it's suggested to apply this basic scheme to modeling of kleptography subliminal channel security.

"Challenge-response" based protocols may be considered as game with $N > 1$ members, one of which is an oracle. Kleptography variant of the protocol includes one more member – Developer, which is considered to be in collusion with one of other members.

Definition 3. (Kleptographic mechanism based on "challenge-response" protocol) Under "challenge-response" based kleptographic mechanism we mean 3-members game (Alice, Bob and Dev) following these rules:

1. Sides Alice and Dev may be in collusion
2. The Alice wait for Bob request, then her response is in format that is preliminary established with Bob. The Alice's goals are:
   (a) Craft response that contains encoded one bit for Dev side
   (b) The encoding must be performed in a way to avoid disclosure by Bob, moreover Bob mustn't know that such transmission were performed
3. Bob sends Alice arbitrary challenge request and receives an answer. The Bob's goal is to detect fact of additional data bit transmission to Dev
4. Dev passively sniffs traffic between Alice and Bob. His goal is to recover hidden bit from Alice side analysing intercepted information

Also, there are such assumptions:

1. Bob isn't in collusion with Alice neither Dev
2. All communicators use standard base cryptographic functions (ciphers, hashes, signatures, etc.) without back-doors
3. Alice and Dev don't use additional steganography channels based on timing, failures, etc.

From Bob's point of view, the protocol may be represented as a couple $\langle D_t, V, U\rangle$, where V is a set of Bob's requests, U – set of Alice's answers, $D_t : V \times U \to \{0,1\}$ – probabilistic algorithm that is bounded with execution time t, it verifies oracle's response to be comply with preliminary negotiated format.

From Dev side, the couple is extended with $R_t^\omega : V \times U \to \{0,1\}$ – algorithm, which extracts bits from subliminal channel between Alice and Dev.

Definition 4. ("Challenge-response" protocol, formal model) Let's denote "challenge-response" protocol as couple $\langle D_t, V, U, A_t\rangle$, where:
$D_t : V \times U \to \{0,1\}$ – probabilistic algorithm with execution time bound t, that verifies oracle's response for protocol compliance. Each correct pair of request-response is recognized with probability 1, so this is Monte Carlo probabilistic algorithm
V – set of Bob's requests, U – set of Alice oracle's responses
$A_t : V \to U$ – Alice's randomized algorithm without subliminal channel: $\forall v \in V : D_t(v, A_t(v)) = 1$

Definition 5. ("Challenge-response" based subliminal channel, formal model) Let's denote "challenge-response" protocol with subliminal channel couple from model 4 $\langle D_t, V, U, A_t, R_t^\omega, A_t^\omega\rangle$ which is extended with additional parameters R_t^ω and A_t^ω (leakage channel), where D_t, V, U, A_t have the same sense as in model 4:
$A_t^\omega : V \times \{0,1\} \to U$ – randomized algorithm of Alice side with hidden bit leakage: $\forall v \in V, s \in \{0,1\} : D_t(A_t^\omega(v,s)) = 1$, $v \overset{rand}{\in} V, s \in \{0,1\} : P\{R_t^\omega(v, A_t^\omega(v,s)) = s\} > 1/2 + \varepsilon(t)$
$R_t^\omega : V \times U \to \{0,1\}$ – probabilistic algorithm, which decodes message from Alice using secret ω.

Also, there are additional security and undetectability requirements: sets $H = \{\langle v,u\rangle | v \in V, u \in U : u = A_t(v)\}$, $H_0 = \{\langle v,u\rangle | v \in V, u \in U : u = A_t^\omega(v,1)\}$ and $H_1 = \{\langle v,u\rangle | v \in V, u \in U : u = A_t^\omega(v,0)\}$ are pairwise indistinguishable: $H \simeq_t H_0 \simeq_t H_1$

Moreover, there is assumption that Dev isn't able to obtain additional information from algorithm A_t output: $|P\{R_t^\omega(v, A_t(v)) = 0\} - P\{R_t^\omega(v, A_t(v)) = 1\}| < \varepsilon(t)$.

Definition 6. (algorithm equality) Randomized algorithms $A_t, A'_t : \mathfrak{L}_1 \to \mathfrak{L}_2$ with execution time bounded with t are considered to be equal ($A_t = A'_t$), if $P\{A_t(l) \neq A'_t(l)\} < \varepsilon(t)$, $l \overset{rand}{\in} \mathfrak{L}_1$.

Theorem 1. *(Leakage channel existence: necessary conditions) If protocol 4 contains a leakage channel, then $\exists A_t, A'_t : V \to U$, $A'_t \neq A_t$, that $P\{D_t(v, A_t(v)) = 1\} > 1 - \varepsilon(t)$ i $P\{D_t(v, A'_t(v)) = 1\} > 1 - \varepsilon(t)$*

Proof. The proof is by design. Let's consider an algorithm $A_0^\omega(v) \equiv A^\omega(v,0)$ and $A_1^\omega(v) \equiv A^\omega(v,1)$, $v \in V$. Then such theorem's requirements are satisfied:

1. $P\{D_t(A_0^\omega(v)) = 1\} > 1 - \varepsilon(t)$ and $P\{D_t(A_1^\omega(v)) = 1\} > 1 - \varepsilon(t)$. Really, according to definition 5 there is channel undetectability requirement, thus $\varepsilon(t) > Adv(A_1^\omega, A_t) \geq P\{D_t(A_t(v))\} - P\{D_t(A_1^\omega(v))\} = 1 - P\{D_t(A_1^\omega(v))\} \Rightarrow P\{D_t(A_1^\omega(v))\} > 1 - \varepsilon(t)$ (the same proof may be applied to A_0^ω).

2. $A_0^\omega \neq A_1^\omega$. Proof by contradiction: let's suppose $A_0^\omega = A_1^\omega$, then $P\{A^\omega(v,0) = A^\omega(v,1)\} = 1-\sigma, \sigma \in [0, \varepsilon(t))$ according to definition 6. So, $P\{R^\omega(A^\omega(v,s)) = 0\} = P\{R^\omega(A^\omega(v,s)) = 1\} = \frac{1}{2}$ with probability $p = 1-\sigma$. However, $P\{A^\omega(v,0) \neq A^\omega(v,1)\} = \sigma$ that's why $\max_s P\{R^\omega(A^\omega(v,s)) = s\} = \xi, \xi \in (1/2, 1]$. The consequence is that full probability is $\max_s P\{R^\omega(A^\omega(v,s)) = s\} = \frac{1}{2}(1-\sigma) + \xi\sigma = \frac{1}{2} + \sigma(\xi - \frac{1}{2}) \in [\frac{1}{2}, \frac{1}{2} + \frac{\varepsilon(t)}{2})$, this contradict the probabilistic algorithm properties R_t^ω from 5.

Thus, algorithms A_0^ω and A_1^ω are instances for algorithms A'_t and A_t from theorem's requirements, so the theorem is proved by design. $\lhd$

Consequence. Let's suppose $\exists A_t, \forall v \in V : P\{D_t(v, A_t(v)) = 1\} = 1$ and $\forall A'_t : A'_t \neq A_t, P\{D_t(v, A'_t(v)) = 1\} = \sigma < 1 - \varepsilon(t)$. Then no one is able to built in leakage channel. Moreover, when hidden bit is transmitted, the detection probability is $P \geq 1 - \sigma$.

Proof. The consequence of $\exists A_t, \forall v \in V : P\{D_t(v, A_t(v)) = 1\} = 1$ and $\forall A'_t : A'_t \neq A_t, P\{D_t(v, A'_t(v)) = 1\} = \sigma < 1 - \varepsilon(t)$ is $\forall A'_t, A''_t : A'_t \neq A''_t, P\{D_t(v, A'_t(v)) = 1\} < 1 - \varepsilon(t) \cup P\{D_t(v, A''_t(v)) = 1\} < 1 - \varepsilon(t)$, so sufficient requirement for leakage channel absence is satisfied.
Further, let's suppose that such predicates belong to transmission session: $Leak \in \{0,1\}$ – there is a fact of hidden data transmission and $Detect \in \{0,1\}$ – the fact of data transmission have been detected. Then full detection probability is $P = P\{Detect = 1 | Leak = 0\} + P\{Detect = 1 | Leak = 1\}$. However, since $P\{Detect = 1 | Leak = 0\} = 0$ (it's impossible to detect the leakage fact if it isn't happened), then $P = P\{Detect = 1 | Leak = 1\} \geq P\{Detect = 1\} = Adv(A_t, A'_t) \geq |P\{D_t(v, A_t(v)) - P\{D_t(v, A'_t(v))\}| = 1 - \sigma$. $\lhd$

3.3 Nonce generation

This basic scheme demonstrates the external Random Number Generator concept (see Picture 1).

Let's consider some protocol which contains step with nonce transmission (e.g., replay attack countermeasure in authentication protocols). If this sequence is r bits, we have steganography container with exactly the same bandwidth.

Input:

1. Asymmetric cryptosystem with private key space K and public key space Q
2. $Sign : K \times \{0,1\}^* \to B$ – digital signature without randomizing, B – signatures space
3. $Verify : Q \times \{0,1\}^* \times B \to \{0,1\}$ – digital signature verifying algorithm
4. Alice's key pair $(k_A, p_A), k_A \in K, p_A \in Q$
5. $\psi : \{0,1\}^* \to \{0,1\}^*$, $\psi_0 : Time \to \{0,1\}^*$ counter increment and initializing.

Algorithm steps:

1. Alice increments her public counter $ctr_i = \psi ctr_{i-1}$. If counter isn't initializes, she generated it, for example, from time stamp: $ctr_0 = \psi_0(time)$.
2. Alice calculates $nonce = Sign(k_A, ctr_i)$ and sends $ctr_i | nonce$
3. Anybody is able to verify that Alice doesn't use her own entropy sources: $Verify(p_A, ctr_i, nonce) == 1$.
4. If there isn't an equality, Alice is suspected to be used nonce as steganography container

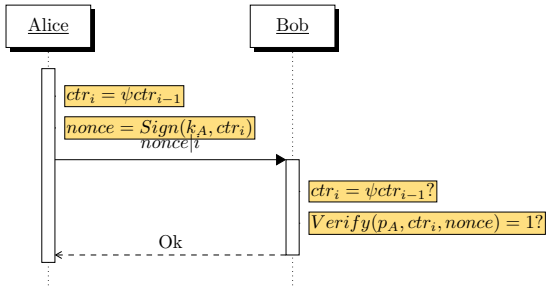

Figure 1: Nonce generation scheme with steganography container detection

Possible malicious scenarios:

1. Attacker guess random sequence before it's publication
2. Attacker restores Alice's private key from captured traffic
3. Developer modifies Alice's protocol implementation to exploit her nonce as steganographic container

In the first case, the attacker must first guess the random sequence R_i of the external generator. Let an intruder intercepts this value at the time of the request by the *Alice* side, but before her nonce is published. Then the attacker needs to obtain $Sign(k_A, R_i)$ from R_i and p_A, i.e., calculate the digital signature without the secret key knowledge, this is reduced to digital signature forgery problem.

In the second case, the attacker intercepts open data: R_i, $Sign(k_A, R_i)$, p_A. Based on these data, he tries to obtain secret key k_A. This can be done either by solving of the Discrete Logarithm Problem (which is reduced to the restoring private key from public one) or to extract it from a published digital signature (which is reduced to the restoring signature key from signature and verification key).

In the third case, the Developer's algorithm fills a nonce with certain message (steganogram) M. The Developer's modification also controls the private key k_A. Consequently, the Developer's modification, having M (signature value) and k_A (secret signature key), and it should be satisfied $Sign(k, ctr_i) = M$. More generalized: the Developer's modification with given M should generate k'_A and R so that $Sign(k'_A, R) = M$ and $Verify(p_A, R, M) = 1$. It requires at least solving signature forgery problem.

Let's formalize this protocol using model 5.

Definition 7. (Nonce generation protocol with steganography container detection)
Nonce generation protocol is a couple $\langle D_t, V, U, A_t \rangle$ according to model 4, where:
V – set of counter's output
U – set of Alice's output
$A_t \equiv Sign(k_A, v)$, $v \in V$ – Alice's algorithm for challenge processing
$D_t \equiv Verify(p_A, v, u)$, $u \in U$

The theorem below will help us to evaluate scheme's kleptography security.

Theorem 2. *Let's suppose:* $\forall v \in V, \forall A_t : A_t(k_A, v) \neq Sign(k_A, v)$, $P\{Verify(p_A, v, A_t(k_A, v)) = 1\} < \varepsilon(t)$ *(i.e., it's impossible in practice to create pair of different signatures for a certain message).*
Then there is no leakage channel in nonce generation protocol 7.

Proof. In the model 7 distinguisher $D_t \equiv Verify(p_A, v, u)$ satisfies sufficient conditions from consequence 1:

1. $\forall v \in V : P\{Verify(p_A, v, Sign(k_A, v)) = 1\} = 1$ by digital signature basic properties
2. $\forall v \in V, A_t(k_A, v) \neq Sign(k_A, v), P\{Verify(p_A, v, A_t(v)) = 1\} = \sigma < \varepsilon(t)$ by our assumptions that it's impossible in practice to create pair of different signatures for a certain message

So, from the consequence 1 and with given assumptions, there is no leakage channel in protocol 4.
Moreover, in case of hidden data transmission, detection probability is $P \geq 1 - \sigma \Leftrightarrow P > 1 - \varepsilon(t)$ if the theorem's 2 assumptions are satisfied.
◁

3.4 SETUP free Diffie-Hellman key agreement protocol

Let's consider secret channel establishment using Diffie-Hellman key agreement protocol between Alice and Bob, which allows to generate random session key pair with proved impossibility of secret key leakage (as in SETUP) (see Picture 2).

Preconditions:

1. Asymmetric cryptosystem (for digital signature) includes private key space $\tilde{K}$ and public key space $\tilde{Q}$
2. Diffie-Hellman asymmetric cryptosystem: G – generator, K, Q – private and public key space, $'\cdot' : K \times Q \to Q$ – agreement function (exponent function)
3. Symmetric cryptosystem (E, D) with key space S, and bijective functions $E : S \times B \to B$ (encryption), $D : \forall s \in S, \forall b \in B, D_s(E_s(b)) = b$ (decryption)
4. $Sign : \tilde{K} \times \{0,1\}^* \to B$ – randomness-free digital signature function, B – signature space
5. $Verify : \tilde{Q} \times \{0,1\}^* \times B \to \{0,1\}$ – digital signature verification function
6. Alice's asymmetric key pair $(k_A, p_A), k_A \in \tilde{K}, p_A \in \tilde{Q}$
7. Bob's asymmetric key pair $(k_B, p_B), k_B \in K, p_B \in Q$
8. $\psi : \{0,1\}^* \to \{0,1\}^*$, $\psi_0 : Time \to \{0,1\}^*$ counter increment and initializing.
9. Cryptographic hash functions $h1 : B \to K$, $h2 : Q \to S$

Algorithm's steps:

1. Alice increments her public counter $ctr_i = \psi ctr_{i-1}$. If counter isn't initializes, she generated it, for example, from time
2. Alice:
   - Generates session private key $q = Sign(k_A, ctr_i | Alice | Bob)$
   - Channel symmetric key: $s = h2(h1(q) \cdot p_B)$
   - Sends public session key, block's id and communicator's ids: $W = h1(q) \cdot G, (W, i, Alice, Bob) \to$ Bob
3. Bob calculates shared symmetric key: $s = h2(k_B \cdot W)$
4. Alice sends her private session key via secure channel: $E_s(q) \to Bob$
5. Bob decrypts key q. He verifies, is the key generated from public counter or no:

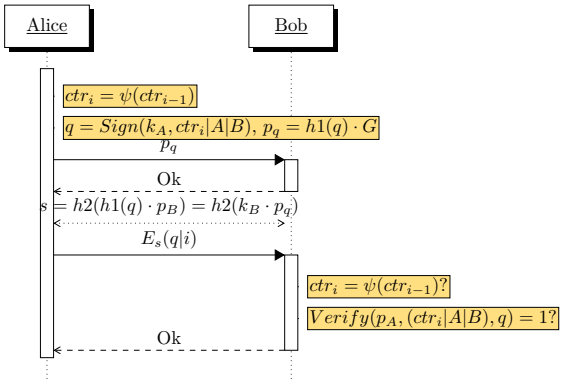

Figure 2: Diffie-Hellman key agreement resistant to SETUP

- Verify $ctr_i = \psi(ctr_{i-1})$ or $ctr_0 = \psi_0(Time)$ (if $i = 0$).
- Verify $h1(q) \cdot G == W$. If it's false, the key is considered to be incorrect, connection should be closed
- Verify $Verify(p_A, ctr_i|Alice|Bob, q) = 1$. If it's false, session private key is generated from unknown source, leakage channel is suspected, connection is untrusted.

Security analysis. Possible malicious scenarios:

1. Attacker forecasts session secret key *Alice*.
2. Attacker restores private key *Alice* analysing captured public traffic.
3. *Bob* as an attacker – he tries to use obtained session key as his own key for communication with third-party
4. Developer modifies *Alice* implementation to get access to data be sent to public channel and use them for leakage channel base (secret key or session private key).

Two first scenarios where considered in 3.3.

In the third case, the *Bob* side receives the private session key *Alice* during the protocol normal working and then tries to use it to key agreement with the other side (say *Dev* side). However, then *Bob* should send along with the public key IDs *Alice* and *Bob*, it's recognized by *Dev* side as an invalid session. When *Bob* sends the identifiers to *Alice* and *Dev*, the *Dev* detects failure in digital signature validation stage: $Verify(p_A, (ctr_i|A|B), q)$

Let the modifier of the Developer pass some kind of encoded secret to an open channel. In the case that additional public messages will be used (in addition to the public key), it may be detected by an external observer that violates the first SETUP property (see Definition 2.1), and therefore the loophole will be exposed.

So, in this case, the Developer's modification can only control a couple of session keys (denote their $(r, R), R = r \cdot G)$), while having access to the private key k_A of the side *Alice*. Additional conditions imposed by the protocol: $Verify(p_A, (S|A|B), r) = 1$,where S is the counter value.This means that the maximal number of possible public keys that can generated by Developer's modification is equal to the number of possible generated public counters and therefore it is impossible in practice to fixate a random secret key.

Let's perform reduction to model 5 for formal proving of leakage channel absence.

Definition 8. (Key agreement protocol without leakage channel)

So, key agreement protocol without leakage channel is a couple $\langle D_t, V, U, A_t\rangle$ from model 4, where:
V – set of possible counter outputs
U – set of possible Alice's outputs
$A_t \equiv h1(q)\cdot G)|e$ – Alice's protocol implementation, $q = h1(Sign(k_A, v)), e = E_s(q)$, $v \in V$
$D_t(v,u) \equiv Verify(p_A, v, getq(u)) * \mathbb{I}(getp(u) = h1(D_{gets(u)}(gete(u)))\cdot G$, $u \in U$, where functions $getq, getp, gets, gete$ are calculated as follow:
$getp(u) = h1(q)\cdot G$,
$gete(u) = e$,
$gets(u) = h2(k_B \cdot getp(u))$,
$getq(u) = D_{gets(u)}(gete(u))$.

Such theorem allows us to evaluate protocol's kleptography security

Theorem 3. *Let's accept such assumptions:*

1. $\forall v \in V$, $\forall A_t : A_t(k_A, v) \neq Sign(k_A, v)$, $P\{Verify(p_A, v, A_t(k_A, v)) = 1\} < \varepsilon(t)$ *(so, crafting pair of different signatures for the same message is impossible in practice).*
2. *Functions $getq, getp, gets, gete$ require negligible execution time*

There is no leakage channel in the Diffie-Hellman key agreement protocol modification 8.

Proof. Let's make sure that sufficient condition for leakage channel absence is satisfied using the theorem's consequence 1. The distinguisher D_t from the model 8 satisfies sufficient from the theorem's consequence 1:

1. $\forall v \in V : P\{D_t(p_A, v, A_t(v)) = 1\} = 1$. Really, algorithm D_t works as follow:
   (a) Obtain $q, h1(q)\cdot G, s, e$ from u using functions $getq, getp, gets, gete$, success probability is $p = 1$.
   (b) Evaluate indicator's value $\mathbb{I}(getp(u) = h1(D_{gets(u)}(gete(u)))\cdot G$. In case of honest protocol, $h1(D_{gets(u)}(gete(u)))\cdot G = h1(D_{gets(u)}(E_s(q)))\cdot G = h1(D_s(E_s(q)))\cdot G = h1(q)\cdot G$, so indicator's value is 1 with probability 1
   (c) Verify signature $Verify(p_A, v, getq(u)) = Verify(p_A, v, q) = 1$ with probability 1 according to digital signature's general properties
2. Le's $A'_t \neq A_t$, i.e. $\exists v \in V : A'_t(v) \neq A_t(v)$, $A_t(v) = h1(q)\cdot G|e$, $A'_t(v) = w\cdot G|e'$. We'll evaluate distinguisher D_t recognition probability A'_t. So, there are such three cases:
   (a) $h1(q)\cdot G \neq w\cdot G \wedge e = e'$. Then $s \neq s', s' = h2(w\cdot k_B\cdot G) \Rightarrow D_{s'}(e) = q' \neq q$, $P\{Verify(p_A, v, q') = 1\} < \varepsilon(t)$ (according to one of the theorem's assumptions)
   (b) $h1(q)\cdot G = w\cdot G \wedge e \neq e'$. Then $q' = D_s(e') \neq D_s(e)$ (because of bijectivity of functions (E, D)), $P\{Verify(p_A, v, q') = 1\} < \varepsilon(t)$
   (c) $h1(q)\cdot G \neq w\cdot G \wedge e \neq e'$. Then $q' = D_{s'}(e')$. If $q' \neq q$, $P\{Verify(p_A, v, q') = 1\} < \varepsilon(t)$. If $q' = q$ probability $P\{Verify(p_A, v, q') = 1\} = 1$. According to the protocol, algorithm D_t verifies $w\cdot G = h1(q')\cdot G$, which contradict the property $h1(q)\cdot G \neq w\cdot G \wedge e \neq e'$

   Thus, maximal probability is $P\{D_t(v, A'_t(v)) = 1\} = \sigma < \varepsilon(t)$

Thus, from the consequence 1 with given assumptions follows that in protocol 4 there is no leakage channel.
Moreover, if there is a fact of hidden data transmission, the detection probability is $P \geq 1 - \sigma \Leftrightarrow P > 1 - \varepsilon(t)$ (based on theorem's 3 assumptions). $\lhd$

Conclusions

In this articles the authors suggested formal model for particular but the most widespread class of SETUP protocols. This model allowed to formulate and prove sufficient conditions for SETUP-free protocol design. Moreover, it were suggested two protocols based on these approaches (nonce generation and SETUP-free improvement of Diffie-Hellman key agreement protocol) that were proved to be resistant against secret key leakage.

It was also suggested the new concept of protocols design which includes usage of digital signature from public counter values instead of usage of random sources. It allows actors of communication verify that their opponents don't use internal random sources and thus detect malicious injection in protocol implementations monitoring potential hidden data leakage channels.

References

[AOS12] Adrian Atanasiu, Ruxandra Olimid, and Emil Simion. On the security of black-box implementation of visual secret sharing schemes. *Journal of Mobile, Embedded and Distributed Systems*, 4(1):1–11, 2012.

[BG07] Côme Berbain and Henri Gilbert. On the security of iv dependent stream ciphers. In Alex Biryukov, editor, *Fast Software Encryption*, pages 254–273, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[YY96] Adam Young and Moti Yung. *The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?*, pages 89–103. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[YY97] Adam Young and Moti Yung. *Kleptography: Using Cryptography Against Cryptography*, pages 62–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.