Bringing Kleptography to Real-World TLS

Adam Janovsky, Jan Krhovjak, Vashek Matyas

Bringing kleptography to real-world TLS

Adam Janovsky[1(✉)], Jan Krhovjak[2], and Vashek Matyas[1]

[1] Masaryk University
adamjanovsky@mail.muni.cz
[2] Invasys, a.s.

Abstract. Kleptography is a study of stealing information securely and subliminally from black-box cryptographic devices. The stolen information is exfiltrated from the device via a backdoored algorithm inside an asymmetricaly encrypted subliminal channel. In this paper, the kleptography setting for the TLS protocol is addressed. While earlier proposals of asymmetric backdoors for TLS lacked the desired properties or were impractical, this work shows that a feasible asymmetric backdoor can be derived for TLS. First, the paper revisits the existing proposals of kleptographic backdoors for TLS of version 1.2 and lower. Next, advances of the proposal by Gołębiewski et al. are presented to achieve better security and indistinguishability. Then, the enhanced backdoor is translated both to TLS 1.2 and 1.3, achieving first practical solution. Properties of the backdoor are proven and its feasibility is demonstrated by implementing it as a proof-of-concept into the OpenSSL library. Finally, performance of the backdoor is evaluated and studied as a tool for side-channel detection.

Keywords: Asymmetric backdoor · Cryptovirology · Kleptography · TLS

1 Introduction

Tamper-proof devices were proposed as a remedy for many security-related problems. Their advantage is undeniable since they are protected from physical attacks and it is difficult to change the executed code. However, they inherently introduce a trust into the manufacturer. It was shown that such devices are theoretically vulnerable to the presence of so-called subliminal channels [20]. Such channels can be used to exfiltrate private information from the underlying system covertly, inside cryptographic primitives. As a consequence, malware introduced by a manufacturer or a clever third-party adversary can utilise subliminal channels to break the security of black-box devices.

This paper concerns kleptography – the art of stealing information securely and subliminally – for the TLS protocol. The field of kleptography was established in the 1990s by Yung and Young [18]. Kleptographic backdoors for many protocols and primitives were proposed ever since. For instance, we mention the RSA

key generation protocol [18] and the Diffie-Hellman (DH) protocol [19]. Using kleptography, an attacker can subvert a target cryptosystem to deny confidentiality and authenticity of transferred data. Thus, it is important to explore the feasibility of kleptographic backdoors for various protocols, alongside with the methods for defeating such backdoors.

Several challenges arise when inventing a kleptographic backdoor. First, one must assure that such backdoor cannot be detected by looking at inputs and outputs of an infected device. Furthermore, the exploited channel is often narrow-band and the computing performance of the device should not be overly affected. Last but not least, one must prove the security of both the original cryptosystem and the encrypted subliminal channel.

Previous work on kleptography in the TLS protocol [9,21] showed that it is possible to utilise a single random nonce to exfiltrate session keys. Both backdoors exploit a random field inside *ClientHello* message, 32-byte nonce that is sent to server by a client. The proposal [9] was rather a sketch of an asymmetric backdoor and it lacked few key properties. The work [21] was an important theoretical result and proven asymmetrical backdoor for the TLS protocol. Nonetheless, it remains impractical to implement. Also, neither of the papers addressed TLS of version 1.3. More insight into the related work follows in Section 5.

In this work we make the following contributions:

- We modify the backdoor [9] to achieve better security of the backdoor and also idistinguishability in a random oracle model.
- We prove that our proposal is an asymmetric backdoor for all versions of the TLS protocol, including TLS 1.3.
- We implement the backdoor as a proof-of-concept into the OpenSSL library, confirming its feasibility.
- We evaluate the performance of the backdoor and discuss its detectability.

The remainder of the paper is organized as follows. Section 2 gives basic background on kleptography. In Section 3 we show a design of our backdoor. Section 4 comments on how we implemented the backdoor and gives the exact results of our performance tests. It also shows how a timing channel can be used to detect our backdoor. Section 5 reviews related work and, finally, Section 6 concludes the paper.

2 Kleptography background

The work on kleptography utilises cryptology and virology and naturally extends the study of subliminal channels [20]; those are further encrypted and embedded

into the devices, creating so-called asymmetric backdoors. As of 2018, secretly embedded backdoor with universal protection (SETUP) is a supreme (and only) tool in the field of kleptography. One could therefore say that kleptography studies development of asymmetric backdoors and possible defenses against them at the same time. Kleptography concerns black-box environment exclusively, as in white-box setting scrutiny allows to detect such channel. The aim of this section is to introduce necessary techniques that are involved in an asymmetric backdoor design. We begin with a formal description of an asymmetric backdoor adopted from [20].

Definition 1. *Assume that C is a black-box cryptosystem with a publicly known specification. A SETUP mechanism is an algorithmic modification made to C to get C' such that:*

1. *The input of C' agrees with the public specifications of the input of C.*
2. *C' computes efficiently using the attacker's public encryption function E (and possibly other functions) contained within C'.*
3. *The attacker's private decryption function D is not contained within C' and is known only by the attacker.*
4. *The output of C' agrees with the public specifications of the output of C. At the same time, it contains published bits (of the user's secret key) which are easily derivable by the attacker (the output can be generated during key-generation or during system operation like message sending).*
5. *Furthermore, the output of C and C' are polynomially indistinguishable to everyone except the attacker.*
6. *After the discovery of the specifics of the SETUP algorithm and after discovering its presence in the implementation (e.g. reverse engineering of hardware tamper-proof device), user (except the attacker) cannot determine past (or future) keys.*

Consider that an asymmetric backdoor itself can be a subject of cryptanalysis. That is why the resulting subliminal channel must be encrypted according to good cryptographic practice. To keep the notation unambiguous, we call the person who attacks the backdoor an *inquirer*. We say that an asymmetric backdoor has (m, n) leakage scheme if it leaks m keys/secret messages over n outputs of the cryptographic device. The desired leakage bandwidth that asymmetric backdoor should achieve is (m, m), meaning that the whole private information is leaked within one execution of the protocol. Further, a publicly displayed value that also serves as an asymmetric backdoor is denoted a *kleptogram*.

2.1 Example of asymmetric backdoor

The paper continues with an example of RSA key generation SETUP [18] to illustrate the concept of asymmetric backdoors. The backdoor allows for efficient factorization of RSA modulus by evil Eve. First, Eve generates her public RSA key (N, E) and embedds it into the contaminated device of Alice together with a subverted key-generation algorithm:

1. The device selects two distinct primes p, q, computes the product $pq = n$ and Euler's function $\varphi(n) = (p-1)(q-1)$.
2. The public exponent is derived as $e = p^E \pmod N$. If e is not invertible modulo $\varphi(n)$, new p is generated.
3. Private exponent is computed as $d = e^{-1} \pmod{\varphi(n)}$.
4. Public key is (n, e), private key is (n, d).

After obtaining the kleptogram e contained in public key (n, e), Eve can use her private key (N, D) to compute

$$e^D = p^{ED} = p \pmod N.$$

Thus, Eve can factorize the modulus n, and compute the private exponent d just by eavesdropping the public key. The reader may notice that the backdoor requires e to be uniformly distributed on the group (otherwise the backdoor can be detected), which leaves it unsuitable for real use. Yet, this toy example illustrates the concept of asymmetric backdoors beautifully. It is not difficult to prove that the backdoor fullfils all conditions of SETUP, if e is to be picked uniformly in the clean system. Also, we note that this backdoor exhibits the ideal leakage bandwidth (m, m). However, this backdoor lacks perfect forward secrecy. Indeed, when the attacker's private key (N, D) is compromised, an inquirer can factorize all past and future keys from the particular key generating device.

2.2 Kleptography in the wild

Recall that an asymmetric backdoor is a modification of already established algorithm. Detection of such modification therefore proves its malicious nature. In contrast, when an algorithm is designed to be kleptographic initially, its malevolence cannot be decided so easily. To illustrate this aspect, we briefly revisit the `DUAL_EC_DRBG` pseudorandom number generator invented by the NSA[3]. The generator was standardized in NIST SP 8000-90A [2]. Later, a bit predictor with advantage 0.0011 was presented in [8]. Despite this serious flaw we concentrate on a different problem. In particular, the paper [2] shows potential kleptographic tampering. To be exact, the generator requires two constants on an elliptic curve,

[3] National Security Agency.

i.e., $P, Q \in E(\mathbb{F}_p)$, and the security of the internal state relies on the intractability of discrete logarithm problem for these constants. Consequently, if one is able to find scalar k such that $P = kQ$, they are able to compute the inner state of the generator efficiently based on the output. This naturally breaks the security of the generator. Despite the fact that arbitrary P, Q can be used for the generator, NIST standard forces the use of fixed constants with unknown origin. Naturally, NSA is alleged to provided the backdoored constants. The practical exploitability of backdoored constants was shown in [17]. However, one cannot prove nor disprove that the constants for the standard are backdoored except for the NSA. This aspect suggests that not many SETUPs are likely to appear in the wild, but rather delicate modifications of otherwise secure algorithms are expected, such that their sensitivity to efficient cryptanalysis can be viewed as a coincidence.

3 Attack design

Our proposal is based on [9] by Gołębiewski et al. However, several drawbacks are eliminated by our construction, and properties of the backdoor are treated more rigorously. The needed improvements w.r.t. the proposal [9] were:

- To achieve indistinguishability of kleptogram from random bit string,
- to ensure that reverse-engineering of the infected device will not compromise security of any session,
- to allow recovery of master secret to attacker even if she misses to eavesdrop some sessions.

An additional goal was to minimize the computational overhead introduced by the backdoor to avoid possible detection by timing analysis.

3.1 Backdoor description

During the TLS handshake, assuming no pre-shared key is involved and a new session is to be established, all traffic keys are derived from a pre-master secret and publicly available values. Thus, for an attacker, it suffices to obtain the pre-master secret to decrypt whole session. Additionally, the pre-master secret can be derived via DH method, eventually via RSA method in the case of TLS 1.2 and lower [4] (The removal of the RSA key exchange method from TLS 1.3 makes it more difficult to debug or inspect encrypted connections for the industry

[4] By DH method we refer to DH modulo prime. Nowadays, Diffie-Hellman over elliptic curves (ECDH) is mostly used in TLS connections. We stick to the modular case, even though our method can be translated to elliptic curves easily.

– for example in datacenters of intrusion detection systems. This led to two RFC drafts [10, 11] that would mitigate this issue. The former allows for opt-in mechanism that allows a TLS client and server to explicitly grant access to the TLS session plaintext. The latter relies on introducing static DH key exchange method to TLS 1.3. Naturally, both drafts inherently weaken the TLS 1.3 protocol and did not become a part of the final TLS 1.3 RFC. A question arises, whether the stated motivation behind introducing such drafts was honest, as debugging and inspection of traffic is possible even with ephemeral DH, only at a cost of adjusting infrastructure.). In the case of DH method, a shared secret is established between the server and the client. In the case of RSA method, server's certificate is required and used to encrypt random bytes generated on the client device. Those bytes then serve as the pre-master secret. We exploit the 32-byte random nonce sent by the client during *ClientHello* message to derive a secret only the attacker can obtain. We further sanitize that secret and use it as a seed during the function that creates the client's contribution to the pre-master secret.

We begin with the presentation of the original kleptographic construction by Gołębiewski et al. The authors suggest to hardcode a DH public key $Y = g^X$ on an infected device. During the first handshake on the device, a random value k is selected and g^k is published as the *ClientHello* random nonce. During subsequent executions, the *ClientHello* random nonce is not subverted, but the PMS is then derived deterministically from $H(Y^k, i)$, where H denotes a hash function and i is a counter to ensure that the secrets will differ across sessions. Notice that when an attacker fails to eavesdropp the first handshake, she will not be able to recover any of subsequent sessions. At the same time, if an inquirer manages to capture the first handshake and any of k or X, it allows the inquirer for a decryption of all previous and subsequent sessions. Also, the value Y^k must be stored in non-volatile memory on the infected device and is prone to reverse engineering, thus violating condition 6 of SETUP. Last but not least, the published nonce g^k can be distinguished from a random bit string since it is an element of a group, see [6] – this violates condition 5 of SETUP. Our improvements aim to eliminate all of the presented drawbacks.

The exact design of our proposal is as follows. Prior to the deployment, a designer generates a DH key pair on the X25519 curve, denoted $Y = g^X$. The public key Y is then hard-coded into the infected device, together with the 128-bit key for AES, denoted K and the counter. The initial value of the counter is 1 and is incremented by 2 after each execution. Suppose that the infected device connects to the server and the handshake is initiated. When construction of the *ClientHello* message is triggered, the infected device generates 32 random bytes denoted k and computes the public key g^k. The value g^k is then encrypted with AES-CTR into $C = E_K(g^k)$ and published as a kleptogram inside the *ClientHello* random nonce. Meanwhile, value $S = Y^k$ is derived on the device as a shared secret between the attacker and the device. When the attacker eavesdropps the value C, she is able to derive $g^k = D_K(C)$ and then $S = g^{kX}$ using

Algorithm 1: Generate kleptogram and seed

Input: A public key Y, AES-CTR key K with counter
Output: The kleptogram C and seed S
$k \leftarrow 32$ random bytes
$C \leftarrow E_K(g^k)$
$S \leftarrow Y^k$
delete value k securely
return (C, S)

Algorithm 2: Generate pre-master secret

Input: Key exchange method, DH parameters and public key of server if needed, seed S
Output: Pre-master secret PMS
if *key exchange method is RSA* **then**
  PMS $\leftarrow$ PRF(S, 46 bytes)
else
  $l \leftarrow$ length of DH prime in bits
  $Z \leftarrow$ DH public key of server
  $x \leftarrow S$
  do
    $x \leftarrow$ PRF(x, l bits)
  while $x = 0$ *or* $x = 1$ *or* $x \geq p$
  PMS $\leftarrow Z^x$
end
return *PMS*

her private key X. After obtaining S, the attacker can replicate the computation of the infected device. When the pre-master secret is to be derived, we differentiate two cases:

1. If the RSA method is used, the value S is stretched to 46 bytes by the TLS 1.2 pseudorandom function (PRF) and sent as the pre-master secret.

2. If the DH method is used, the server first sends the DH parameters to the client, including the prime p. The value S is then stretched by the TLS 1.2 PRF to the string of the same length as prime p. This bit string is checked to fulfill requirements for DH private key (not being 0, 1 or $\geq p$) and is used as the client's private key. If the requirements are not met, the output of PRF is repeatedly used as an input to PRF until proper key is generated.

Once the pre-master secret is generated, the handshake continues ordinarily.

The backdoor is described by Algorithms 1 and 2. Algorithm 1 generates the *ClientHello* random nonce and the seed S. The latter is further processed by Algorithm 2 to derive the pre-master secret.

The paper [14] proves that the counter mode (CTR) is polynomially indistinguishable from random bit string on the assumption that the underlying cipher is a pseudorandom function (PRF). This holds when the value of the counter never repeats. We have selected the AES in the CTR mode with a key of 128 bits to achieve indistinguishability. Some properties of this selection must be further discussed. First, NIST recommends [2] to limit the number of calls of pseudorandom number generator (PRNG) keyed with hard-coded value to 2^{48} blocks. Since AES-CTR is essentially a PRNG, this recommendation should be respected. Consider that birthday collisions are likely to appear only after 2^{64} bits of output, so they are trivially treated by the NIST recommendation. Since the backdoor uses two blocks of AES-CTR for one handshake, this limits the functioning of the backdoor to 2^{47} handshakes. It is emphasized that once the backdoor is reverse engineered and the symmetric key is obtained, the indistinguishability is broken.

3.2 Properties of SETUP proposal

Theorem 1. *Under the following assumptions, our proposal is a SETUP:*

- *A random oracle is used to generate the values k and to sanitize the values C instead of a TLS PRF.*
- *AES is a random permutation.*
- *Computational DH assumption holds.*

Proof. The reader can easily verify that properties 1-3 of SETUP hold for our backdoor. To prove property 4, we show how the attacker can obtain the seed S by eavesdropping on the handshake traffic. Notice that once the seed S is known, anyone can replicate the computation of the device that leads to the pre-master secret.

When the attacker obtains the *ClientHello* nonce, she can decrypt it with the AES key K, obtaining the public key of the device g^k. The attacker can further utilise her private key X to compute the shared value $S = g^{kX} = Y^k$. Consider that when the DH key exchange method is used, the attacker must also eavesdrop the public key of the server and the parameters of the exchange; but those are sent in plaintext. To conclude, property 4 holds as well.

We proceed with the property 5. If AES is a random permutation, then AES-CTR produces ciphertext indistinguishable from uniformly distributed bit string as shown in [14]. Thus, one cannot distinguish between the kleptogram C and random bit strings (unless the collisions occur, which was discussed earlier). Further, the resulting pre-master secrets (both for RSA and DH method) are uniformly distributed too, since we utilize the random oracle to sanitize C.

Recall that non-volatile memory of the infected device contains AES key K with the counter and the public key Y. Only the key Y is relevant to the confidentiality

of the shared secret S. Notice that obtaining shared secret g^{kX} from g^k and g^X is equivalent to solving ECDH problem. We therefore conclude that the property 6 holds. □

To summarize, properties 1, 2, 3, 4 hold unconditionally for our proposal. The property 6 requires a computational DH assumption. Moreover, the property 5 requires a random oracle and AES to be a random permutation. This seems sufficient, as for the practical deployment, speed is more pressing than provable indistinguishability.

Regarding the perfect forward secrecy, an inquirer is not able to recover past session secrets if she obtains the private key of the device, k. However, after obtaining the key X, the inquirer can break all past (and future) sessions.

4 Attack implementation

We have implemented the asymmetric backdoor into the OpenSSL library of version 1.1.1-pre2. Choosing this library allowed us to reveal whether the backdoor can pose as a regular malware, without the requirements of a black-box environment. When one designs malware in black-box setting, she is allowed to change both implementation and header files of the infected library. On the contrary, only the compiled binaries are infected in case of regular malware. The OpenSSL does not expose many low-level functions from a cryptographic library to high-level functions that are used in the TLS handshake. Our work shows that the proposal can be embedded into the compiled binary, leaving the header files untouched.

The pre-release version of the library was chosen because it provides certain functions for computations on the X25519 curve not available in previous releases. The X25519 is implemented in a different way than other elliptic curves in OpenSSL and older releases did not allow for the creation of specific keys on this curve; only random keys could be created. We decided to expose some low-level functions for direct use to achieve simpler implementation. This resulted into modification of the header files. Nevertheless, high-level interfaces could be used instead and the backdoor could be deployed as a compiled library. We faced no serious obstacles that would prevent the backdoor installation to the library.

4.1 Attack detection

We have also studied the possible detections of the infected library via side channels. As we worked in the desktop environment, we limit our attention to the

timing channel. Nonetheless, a power side channel could be a viable detection mechanism on different platforms. Several code snippets of both infected and clean version of OpenSSL were isolated and their performance was evaluated and compared. This creates a possible detection mechanism of the backdoor, yet, with certain limitations. As expected, the backdoor performs slower than the clean version. Nevertheless, this does not necessarily create a distinguisher. Suppose that all devices of a certain kind are infected. Then there exists no reference to how the uninfected version should perform. The inquirer must therefore somehow guess the expected performance and measure deviations based on this estimate. Also, the library could be used on various hardware and outperform clean versions when running on faster hardware.

Three code snippets were measured for the infected algorithm. In particular, those were the RSA pre-mater secret generation snippet, the DH private key generation snippet, and the *ClientHello* nonce generation snippet. The last snippet was measured in two versions. The first version contained only one exponentiation (computation of the public-key presented as kleptogram). The second version was expanded by shared-secret derivation between the attacker and the device. As the backdoor does not require any initialization except for loading the AES counter (other keys can be hard-coded in the binary), this aspect was ignored in the experiments.

Code snippet	Average computation time in μs
Timer overhead	0.312
g^k on X25519	171.132
ClientHello clean	4.726
ClientHello subverted	176.293
ClientHello subverted[a]	246 843
RSA PMS clean	4.887
RSA PMS subverted	11.185
DH private key gen. clean	3178.587
DH private key gen. subverted	3218.496
TLS context builder	374.5

Table 1. Average execution times of code snippets.

[a] Version with the shared secret precomputation.

4.2 Average execution times

Our measurements show that the whole subverted version runs by 0.248 ms (0.282 ms) slower than the clean version when RSA (DH) key exchange method

is used. The subverted RSA key exchange method runs slower by the factor of 2.28 over the clean version. On the contrary, the subverted DH key exchange method runs slower only by the factor of 1.01. This is because in the case of RSA, the newly introduced computations take relatively much more time to the overall key exchange method cost. It also can be seen that such an increase in time cannot be spotted just by using the device. The interaction over the network creates the opportunity for obfuscating the computation times. The exponentiations, or even parts of them, could be precomputed once the handshake is initiated, stored, and only loaded from memory when needed. The more complex the underlying protocol is, the larger is the space for obfuscations. It is also questionable whether the inquirer will be able to isolate the corresponding snippets on a tamper-proof device to obtain precise measurements.

To conclude, the timing-channel method is not reliable and most likely could be evaded by a skilled adversary. Nevertheless, the proposal can be detected when a clean version of the OpenSSL is at hand and benchmarking is available on the same hardware on which the suspected version is running. As the highest increase in time is seen when the *ClientHello* nonce is generated, this could be the sweet spot for malware detection. Recall that the execution time of the *ClientHello* nonce function should correspond to the time in which the library generates 32 random bytes. If the function takes substantially larger amount of time, the backdoor is likely to be present.

We do not release the source code for the reason that it could be easily misused as a malware.

5 Related work

Our work is based on the concept [9]. In contrast to the original backdoor, our solution cannot fullfils the conditions 5 and 6 of the SETUP mechanism. The paper [22] by Young and Yung presents how to generate shared secret with ECDH that is polynomially indistinguishable from random bit string. Furthermore, they also mention its applicability to the TLS protocol and provide first asymmetric backdoor for TLS. However, the proposal is rather impractical as to execute a single backdoored handshake more than 300 ECDH key exchanges are required. Injective mappings of strings on elliptic curve points [1,6] could have interesting applications for kleptography, as their inversion could map ECDH keys to strings that are polynomially indistinguishable from random strings. Regarding the detection of backdoors via side channels, the paper [12] presents a method that studies variance in execution times of functions which might reveal newly introduced exponentiations to the protocol – a common facet of kleptography.

In recent years, major advances came in the field of defenses against kleptographic adversaries. Most of them were published in [16]. The work achieves a

general technique for preserving semantic security of a cryptosystem, if put into the kleptographic setting. Also, the paper classifies already proposed defenses into three categories:

- Abandoning the randomness in favour of deterministic computation [3–5],
- use of a trusted module that can re-randomize subverted primitives [7,13],
- hashing the subverted randomness [15].

6 Conclusions

TLS is an essential protocol for securing data on the transport layer. As such, TLS is omnipresent in an era of computer networks, having applications in https, VPN, payment gateways and many others. The widespread use of TLS motivated us to study its vulnerability in the kleptographic setting. We aimed to answer whether a kleptographic backdoor can be practically implemented into the TLS libraries.

Our efforts resulted into a design of an asymmetric backdoor for all versions of the TLS protocol. Such backdoor can be used to exfiltrate session keys from a captured handshake by a passive eavesdropper, leading to a denial of confidentiality and authenticity of the whole session. We also demonstrated that it is fairly simple to implement the backdoor into an open source TLS library while maintaining a reasonable performance of the library. We stress that to install our backdoor, an adversary must have access to the target device. In such cases, other dangerous scenarios arise – we mention ransomware as an example. However, the important property of our backdoor is that it may stay unnoticed for a long time on the target device. Also, it may be endorsed into a particular hardware by its manufacturer or organizations with sufficient resources. We also showed that timing analysis may prove as an effective defense, depending on the powers of an inquirer.

For future work, we suggest to study whether an effective defense could be derived for TLS on a protocol level, for instance, in the form of a protocol extension. Regarding the offensive techniques, if mappings [1,6] could be combined with a cryptographic key, they would allow for ECDH secrets indistinguishable from a random noise.

References

1. Aranha, D.F., Fouque, P.A., Qian, C., Tibouchi, M., Zapalowicz, J.C.: Binary elligator squared. In: Selected Areas in Cryptography – SAC 2014. LNCS, vol. 8781, pp. 20–37. Springer, Cham (2014)

2. Barker, E.B., Kelsey, J.M.: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Tech. rep. (2015)
3. Bellare, M., Hoang, V.T.: Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In: Advances in Cryptology – EUROCRYPT 2015. LNCS, vol. 9056, pp. 627–656. Springer, Berlin, Heidelberg (2015)
4. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security – CCS '15. pp. 1431–1440. ACM, New York (2015)
5. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Advances in Cryptology – CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer-Verlag, Berlin, Heidelberg (2014)
6. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security – CCS '13. pp. 967–980. ACM, New York (2013)
7. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls – secure communication on corrupted machines. In: Advances in Cryptology – CRYPTO 2016. LNCS, vol. 9814, pp. 341–372. Springer-Verlag, Berlin, Heidelberg (2016)
8. Gjøsteen, K.: Comments on dual-ec-drbg/nist sp 800-90, draft december 2005. Tech. rep.
9. Gołębiewski, Z., Kutyłowski, M., Zagórski, F.: Stealing secrets with SSL/TLS and SSH – kleptographic attacks. In: Cryptology and Network Security – CANS '06. LNCS, vol. 4301, pp. 191–202. Springer-Verlag, Berlin, Heidelberg (2006)
10. Green, M., Droms, R., Housley, R., Turner, P., Fenter, S.: Data Center use of Static Diffie-Hellman in TLS 1.3. RFC Draft (2017), https://tools.ietf.org/html/draft-green-tls-static-dh-in-tls13-01
11. Housley, R., Droms, R.: TLS 1.3 Option for Negotiation of Visibility in the Datacenter. RFC Draft (2018), https://tools.ietf.org/html/draft-rhrd-tls-tls13-visibility-01
12. Kucner, D., Kutyłowski, M.: Stochastic Kleltography Detecion. In: Public-Key Cryptography and Computational Number Theory. pp. 137–149. De Gruyter (2001)
13. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Advances in Cryptology – EUROCRYPT 2015. LNCS, vol. 9056, pp. 657–686. Springer, Berlin, Heidelberg (2015)
14. Rogaway, P.: Evaluation of some blockcipher modes of operation. Tech. rep., Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan (2011)
15. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: Clipping the power of kleptographic attacks. In: Advances in Cryptology – ASIACRYPT 2016. LNCS, vol. 10032, pp. 34–64. Springer-Verlag, Berlin, Heidelberg (2016)
16. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic semantic security against a kleptographic adversary. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security – CCS '17. pp. 907–922. ACM, New York (2017)
17. S. Checkoway, et al.: On the practical exploitability of dual ec in tls implementations. In: SEC'14 Proceedings of the 23rd USENIX conference on Security Symposium. pp. 319 – 335 (2014)

18. Young, A., Yung, M.: The dark side of "black-box" cryptography or: Should we trust capstone? In: Advances in Cryptology – CRYPTO '96. LNCS, vol. 1109, pp. 89–103. Springer-Verlag, Berlin, Heidelberg (1996)
19. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Advances in Cryptology – EUROCRYPT '97. LNCS, vol. 1233, pp. 62–74. Springer, Berlin, Heidelberg (1997)
20. Young, A., Yung, M.: Malicious Cryptography: Exposing Cryptovirology. Wiley, Hoboken, NJ (2004)
21. Young, A., Yung, M.: Space-efficient kleptography without random oracles. In: Information Hiding: 9th International Workshop, IH 2007. LNCS, vol. 4567, pp. 112–129. Springer-Verlag, Berlin, Heidelberg (2007)
22. Young, A., Yung, M.: Kleptography from standard assumptions and applications. In: Security and Cryptography for Networks. LNCS, vol. 9841, pp. 271–290. Springer International Publishing (2010)