Side Channel Attack by Convolutional Neural Network

Haobei Song
20629096
University of Waterloo

April 22, 2018

Abstract

In Cryptography, side channel attack is implemented by exploiting the measurement taken on the computing devices of a cryptosystem. Classic techniques involving statistical analysis such as template attack have been shown effective by analysing the measurement from side channels. However, this classic method does not scale well and the hypothesis space is limited by human crafted hyper-parameters. To overcome the limitation, machine learning techniques were introduced to extract information effectively from large dimmensional raw dat and make predictions in a larger hypothesis space. More efficent deep learning techniques are under experiment and results demonstrate better performance than machine learning techniques such as Support Vector Machine (SVM) and Random Forest (RF). In this paper, the feasibility of side channel attack by Convolutional Neural Network (CNN) is analyzed in a theoretic manner and compared with other techniques.

1 Introduction

Side Channel Attack is a category of attacks based on the analysis of physical measurment on cryptographic devices. This includes timing attack, which exploits the difference of time in executing cryptographic instructions, power comsumption, electromagnetic waves, etc. These side channel measurements generally leak information about the secret keys themselves in most of the cryptographic protocols and thus the secret key could be recovered much more efficient than exhaustive search, which can be proved secure or to be the most efficient algorithm in breaking the protocol under reasonable assumptions. This could have a catastrophic impact in real world as most of the protocols being deployed are vulnerable to side channel attacks.

Side Channel Attacks can be divided into two classes

Profiling Side Channel Attack gives the adversary more powerful capability than Non-Profiling SCA as this kind of attack enables the adversary to procure a copy of the physical devices which perform cryptographic computation. Most of the attacks such as template attack, SVM, RF, CNN, etc. under research fall into this category and it has shown that the secret key can be recovered with high probability for many popular symmetric key encryption protocols such as Triple-DES, AES.

Non-Profiling Side Channel Attack is a much weaker version of SCA, for that the adversary are constrained to be able to passively collect data leaked from the targeted devices which perform cryptographic computation.

Though people has been long before concerned with timing attacks, it is not until recently that people start to realize the profound implications of other the more and more effective SCA techniques, better formed countermeasure has been proposed.

Hiding countermeasures add complementary instructions to make the computations performed on target physical devices consistent with respect to time, power consumed, etc. A familiar example is the Montgomery ladder method to do elliptic curve point multiplication, which though is not the most efficient algorithm but one taking fixed amount of time for any two rational points randomly selected in the elliptic curve group.

Masking Countermeasures resemble the idea of perfect secret sharing where sensitive variables derived from secret key (which might leak some information exploited by side channel measurement) are divided into n shares (n is directly related to the security parameter). No adversary under certain assumption could recover the information in the sensitive variables with any (n-1) shares.

2 Profiling Side Channel Attack Techniques

The rest of the discussion will focus on profiled side channel attacks as this class of attacks is considerably efficient in breaking real world protocols.

3 Template Attack

In a Symmetric Key Encryption (SKE) setting, the secret key is $k \in \{0,1\}^n$ where n is the security parameter. Consider one side channel measurement as the trace of power consumption in the physical cryptographic device while doing the encryption or decryption. Denote the trace as a N dimensional vector $T^k \in R^N$. The current state-of-art template attack models the trace as a multivariate guassian distribution with parameters associated with different key k being used. namely

$$P(T^k)|\mu(k),\Sigma(k)) = \frac{1}{\sqrt{(2\pi)^n|\Sigma(k)|}} e^{-\frac{1}{2}(T^k-\mu(k))\Sigma(k)^{-1}(T^k-\mu(k))^T}$$

Where $\mu(k)$ is mean value of key k and $\Sigma(k)$ is the covariance matrix of key k. The sample estimate of $\mu(k),\Sigma(k)$ is

$$\hat{\mu(k)} = \frac{1}{|S|}\sum_{i=1}^{|S|} T_i^k$$

$$\hat{\Sigma(k)} = \frac{1}{|S|-N}\sum_{i=1}^{|S|}(T_i^k - \hat{\mu}(k))^T(T_i^k - \hat{\mu(k)})$$

The 'training' phase of template attack is basically the calcution above and the 'prediction' algorithm is as following assuming p traces are abtained for the same secret key

$$\hat{k} = \underset{k}{\operatorname{argmax}} \prod_{j=1}^{p} P(T^j|\mu(k),\Sigma(k))$$

This technique only takes into concern Gaussian models, which results in small hypothesis space compared to kerneled SVM and the number of parameters (in $\mu(k)$ and $\Sigma(k)$) grows quatratically as N goes large. Therefore feature extraction technique such as Principle Component Analysis (PCA) is used to effectively reduce the dimensionality of the raw data. However PCA necessarily loses information about the underlying sensitive varables during the dimension reduction pre-processing, template attack can not scale well.

4 Machine Learning

Machine learning generally scale well with large amount of data and can be applied directly to the raw data to extract maximum amount of information related to sensitive variables or the secret key itself. With kernel trick, the hypothesis can be much larger than the multivariate Gaussian model assumed by classic template attack because kernel method uses non-linear transformation to map the raw data to a larger dimension, which enbles the linear model to model much complicated non-linear models.

5 Kerneled Support Vector Machine

Kerneled Support Vector Machine is the state-of-art machine learning algorithms which performs surprisingly well on many difficult tasks. It takes the name support vector from the effective data points which are called support vectors during the training phase. Since only the support vectors are taken into account in the training, this technique simplifies the computation tremendously

as otherwise the computation could be really expensive especially when kernel method is applied since the computation complexity is cubic in the number of data points taken into account.
Still it suffers from the same problem as the dimension of the feature vectors becomes large. This can not be circumed as to train more complicated model requires larger dimension of feature vectors. Deep learning was then introduced to learn a much more complicated attack model.

6 Deeplearning

7 Definitions

7.1 Signature Scheme

A signature scheme is a triple of polytime probabilistic algorithms (PPT) $\Sigma = (K, G, V)$, and they are key generation, signature generation and signature verification respectively. K is given the security parameter n and generates a public key Y and private key x. G is the signer which given private key x and an arbitrary plaintext message m, outputs a signature S. The verifier V can, given public key Y, plaintext message m and signature S, accept or reject the signature. A correct signature scheme should always have that the verifier can accept a signature signed by the true signer with failure rate negligible and reject a forged signature with success probability negligible.

7.2 Forger

A forger of a signature scheme $\Sigma = (K, G, V)$ is a PPT F, which given public key Y, a signature S and an internal state X, output a message m, state X and a signature R or a request, also denoted as R for a signature of a message m_i.

A forger game for a specific forger F is a game consisting of counterplay of the following:
Initially, the signer calls K to output a public key which is revealed to the forger and a private key x which it keeps secret.
The forger (which knows the public key Y) starts from an initial state X_0 and output a message m_i, a state X_i and R_i a signature or a request for signature. For $i \geq 1$. If R_i is a request for a signature, then the signer correctly sign the message m_i and call the forger with input the signature S_i for m_i and the state of the forger X_i. If R_i is a signature, then the forger outputs the signature and the game is over.
An additional constraint is that when the game ends the message for which the forger forge the signature for was not queried to the signer. Otherwise this forger would not be useful.
A signature scheme Σ is (p,Q,t) secure if it does not have a (p,Q,t) forger which could win the forger game in at most Q rounds performing t computations with

probability at least p. When Q equals 0 the forger is said to be passive and active otherwise. A selective forger is in addition given a message to be signed before the forger game starts and only succeed until the forger correctly forges a signature for such message which of course is not queried to the signer. In contrast the forger defined previously is called an existential forger.

Signature Security A signature scheme is (p,Q,t)-secure against an existential forgery if there exists no (p,Q,t)-forger and is (p,Q,t,U)-secure against a selective forgery if there exists no (p,Q,t,U)-selective forger where U is a PPT algorithm to generate the initial selective message(challenge message) to be forged. Since a selective forger is more harmful in the sense that a signature scheme breakable by a selective forger also fails existential forger test in the way.

ECDSA Elliptic Curve Digital Signature Algorithm or ECDSA is a tuple $\Sigma = (K, G, V)$ of Key Generation, Signature Generation and Signature Verification algorithms which are all polytime probabilistic algorithms (PPT). Start from the construction of the underlying group ECDSA is built on. n is a prime number of certain bit-length defined by a security parameter. $\mathbb{A}_n$ is a subgroup of an elliptic curve group defined over a finite field with a generator denoted as G of order n. Such an elliptic curve group can be generated efficiently with the above requirements satisfied. The conversion function is just $\mathbf{x}$ which returns the x-coordinate of a point on elliptic curve or more formally:

$$f : \mathbb{A}_n \to \mathbb{Z}_n.$$

A hash function is $h : \{0,1\}^* \to \mathbb{Z}_n$ with more details in the rest of the paper.

Key Generation : A private key is an integer $d \in_R \mathbb{Z}_n \backslash \{0\}$ generated by a pseudorandom generator. The corresponding public key is just $Q = dG$ which could be computed efficiently in the means of double and add.

Signature Generation For a given message M in the message space $\{0,1\}^*$, firstly select a ephemeral key $k \in_R \mathbb{Z}_n$ by a pseudorandom generator and compute $R = kG$.
Call the conversion function to compute $r = \mathbf{x}(R)$. If this returns 0 go back to step 1. Since $\mathbf{x}$ returns zero with negligible probability, this should not be a concern in practice.
Then compute the hash of the plaintext message M to obtain $e = h(M)$. Compute $s = k^{-1}(e + dr) \bmod n$. Similarly, there is a chance that s is zero, But as long as the hash function and pseudorandom generators behave as they are supposed to, this only happens with a negligible probability. The signature on M is (r, s)

Signature Verification To verify a signature (r, s) on a plaintext message M with public key Q, group $\mathbb{A}_n$ and generator of the group G. First check the syntax of the signature and that $r, s \in \mathbb{Z}_n$. Then return ACCEPT if

$$r = f(s^{-1}h(M)G + s^{-1}rQ)$$

and REJECT otherwise.

DSA The Digital Signature Algorithm or DSA differs from ECDSA in the underlying group and the conversion function. Namely

8 Building Blocks

8.1 Discrete Logarithm and Secure Group

The Discrete Logarithm Problem (DLP) is formulated as follows. Let $(\mathbb{G}, *)$ be a group with group operation $*$. A discrete logarithm is, given group elements $G, P \in \mathbb{G}$, the solution (if one exists) x which satisfies that $xG = P$ or equivalently $x = log_G P$. Normally $\mathbb{G}$ is defined in such a way in which a generator always exists and is given beforehand to ensure every group element has a discrete logarithm solution. The Nechaev-Shoup generic model of a group is such a group without any other structural properties to be potentially exploited. In a generic group, the group operation can only be performed by a oracle and the group operation is in some sense random subject to the given group operations. In a nutshell, a generic group is a group with a back-box like group operation. However this is unachievable for any efficient groups. Such a group as a Nechaev-Shoup generic model is also named a *secure group* of strength $|G|/2$. $|G|/2$ comes from the fact that Pollard-ρ-algorithm could find the logarithm solution with probability $\sqrt{\frac{\pi}{2|G|}}$ by birthday paradox.
The secure group $\mathbb{A}_n$ used in ECDSA is a group of order n generated by points on an elliptic curve over a finite field $\mathbb{F}_q$. Depending on the parity of q, the implementation might differ nontrivially.

8.2 Conversion Functions

In the following statement, define $\mathbb{A}_n$ to be the Elliptic Curve group and $\mathbb{Z}_n$ the exponent group. As an example, $x = log_G P, x \in \mathbb{Z}_n, G, P \in \mathbb{A}_n$.
A conversion function is

$$f : \mathbb{A}_n \to \mathbb{Z}_n \tag{1}$$

8.2.1 Almost-Bijectivity

Almost-Bijectivity is defined in terms of *ϵ-clustering.* A conversion function is ϵ-clustered if given $z_0 \in \mathbb{Z}_n, A \in_R \mathbb{A}_n$ The probability that $f(A) = z_0$ is at least ϵ. And a conversion function is *almost-bijective* of strength at least $log_2(1/\epsilon)$ bits if it is not ϵ-clustered.

8.2.2 Almost-Invertibility

An *almost-inverse* of f is defined as a polytime probabilistic algorithm (PPT) $\mathcal{INV}$

Input: $\mathbb{A}_n, z \in_R \mathbb{Z}_n$
Output: A such that $f(A) = z$ with probability at least 1/10

Fact Almost Invertibility implies almost-bijectivity of strength at least $log_2(1/\epsilon)$ bits where $\epsilon > \frac{10}{n}$.

Note The contrapositive of the statement above is easy to prove.
The conversion function for ECDSA is just the x-coordinate of points on the elliptic curve.

$$f : G \in \mathbb{A}_n, \to \mathbf{x}(G) \bmod m$$

Given $z \in \mathbb{Z}_n$, the inverse of z can be calculated as following. First find x in $\mathbb{F}_q$ such that $x \equiv z \pmod{m}$ and x is the x-coordinate of some points on the elliptic curve. Then solve the elliptic equation to recover the y-coordinate (there are two y-coordinates corresponding to the same x-coordinate so one might need to send an extra bit to differentiate them). This method could find the inverse with probability at least 1/8 of the standard implementation in ECDSA. Thus f is almost-inverse which implies f is almost-invertible as well. $x \in$

8.3 Secure Hash Function

A hash function is defined to be a function

$$h : \{0,1\}^* \to \mathcal{H}$$

where the domain $\{0,1\}^* = \{empty\} \cup (\cup_{i=1}^{\infty}\{0,1\}^i)$ is the set of bit strings of arbitrary length.nd the range is $\mathcal{H}$ which is usually of set $\{0,1\}^n$ where n is related to the security parameter.

Preimage Resistant A PPT algorithm $\mathcal{V}$ is a (ϵ, τ) inverter of hash function h if $\mathcal{V}$ given input $e \in_R \mathcal{H}$ outputs a message M in the domain within running time at most τ. And $h(M) = e$ with probability at least ϵ. If no such inverter exists, h is preimage resistant or one-way of strength (ϵ, τ)

Second-Preimage Resistant Let S be an $(\epsilon, \tau, \mathscr{F})$ -second-preimage-finder for hash function h if S on input $M \in_R \mathscr{F}$ outputs $M\prime$ such that $h(M) = h(M\prime), M \neq M\prime$ with probability at least ϵ within running time at most τ. A hash function is said to be second-preimage resistant if no such second-image-finder exists

Collision-Resistant A (ϵ, τ)-collision finder PPT is C such that within running time at most τ S outputs two different messages with the same hash value with probability at least ϵ. And a hash function is collision-resistant if no such collision finders exist.

Zero-Finder-Resistant A hash function h is said to be (ϵ, τ) zero-finder-resistant if no PPT algorithm can find a message M with $h(M) = 0$ within τ and have a success probability greater than ϵ

Zero-Rarity A hash function is zero-rare of strength $(\epsilon, \mathscr{F})$ if for $M \in_R \mathscr{F}$, the probability that $h(M) = 0$ is at most ϵ.
Zero-finder-resistant implies zero-rarity as otherwise one can always try exhaustive search to find enough messages having hash value zero assuming zero-rarity is not true.
I personally don't understand why the author mentioned zero-finder-resistant and zero-rarity here. The ECDSA or ECD only fails when the hash value equals zero with negligible probability.

Uniform Hash Function A uniform hash function is that no PPT algorithm could differentiate the hash function from a true uniformly random function defined on the same domain and range with probability significantly greater than one half.

As an important note, uniformity and collision-resistance together implies preimage resistant but not the opposite way.

Pseudorandom Generators Though pseudorandomness was not mentioned in the original paper in more detail, here is an overview of it. A pseudorandom generator is a random number generator which every time is called with a seed of bit-length n, produces a number x of bit-length m (with $m > n$, or more accurate m is a polynomial of n) such that no PPT algorithms exist which could differentiate it from a random number sampled from a true uniform distribution (the range of the random numbers should agree).

9 Necessary Security Conditions

Intractable Discrete Logarithm Problem If not, then the adversary is able to compute the private key efficiently and undermine the signature scheme by signing any message the same way as a true signer would do (though the ephemeral key might be different).

Almost-Bijective Conversion Function If not, then the conversion function must be ϵ-clustered for a $z_0 \in \mathbb{Z}_n$ and one can construct a selective

$(\epsilon, 1, 0, \mathscr{F})$-forger for any message distribution $\mathscr{F}$ without even making any query to the signing oracle.

$$\text{Select a random } s \in \mathbb{Z}_n$$
$$\text{Output } (z_0, s)$$

This would succeed with probability ϵ as the conversion function is ϵ-clustered.

Preimage Resistance Suppose there exist in inverter $\mathscr{V}$ of strength (ϵ_V, τ_V) of the hash function h. Since with a selected e

$$f(s^{-1}eG + s^{-1}rQ) = f(aG + bQ) \text{ with } a = s^{-1}e, b = s^{-1}r$$
$$r = f(aG + bQ), e = ab^{-1}r, s\prime = rb^{-1}$$
$$\textbf{Output: } (r, s\prime) \text{ on } M = \mathscr{V}(e)$$

The probability that with arbitrarily selected a and b the computed hash value e falls in the range of the hash function is $|\mathscr{H}|/n$. Thus the probability that the above forger succeeds is $\epsilon_V|\mathscr{H}|/n$

Collision-Resistant Hash If not, invoke the collision-finder to find two message M' and M such that $h(M) = h(M')$. Then make a query to the signing oracle to get a signature on M. This signature is also a valid signature for M'. Therefore, there exists an existential forger which succeeds with probability the same as the success probability of the second-preimage-finder.

Second-Preimage-Resistant Hash The same forger for the non-Collision-Resistant hash can break the scheme with the difference being that with a second-preimage-finder, one can find a collision for this specific message and produces the signature as the one queried from the signing oracle on the message that has the same hash value. This forger is a selective forger so it is more harmful if one abandons this requirement for a hash function.

Zero-Finder-Resistant Hash The argument for this property of hash function is similar to the argument for preimage resistance. Set the a in the argument for preimage resistance to 0 and the same proof works here.

In addition, one should always check if the r-value of a given signature is zero, otherwise the above existential forger turns into a selective-forger.

Arithmetically Unbiased k-Value It happens that same ephemeral key was actually used more than once in industry, which completely undermine the secrecy of this signature scheme as one can recover the underlying private key from the signatures generated using the same ephemeral key. Now the stronger statement is that even a slightly biased pseudorandom generator (certain kinds of bias) could be exploited to recover the underlying private key. So a pseudorandom generator to generate ephemeral keys is assumed here.

The Generic Group Model Consider $A = xG, B = yG$, Computing x from A, G is intractable and computing xyG from A, B, G is also intractable thought there might be some information leaked from xy the multiplication of the exponents. This is formulated as Elliptic Curve Diffie-Hellman Problem which no one ever found an efficient solution for. However, this problem might be solved or simplified by exploiting some group properties of certain elliptic group. For the purpose of proving ECDSA, it is reasonable to assume a generic model for the underlying elliptic curve group. More specifically, two operations are provided for any forger to invoke as well the the verifier, namely push and subtract command.

Push Command For any input A denoted as $A_{m+1} \in \mathbb{A}_n$ in the $m + 1$th invocation store the exponent $z_{m+1} + x_{m+1} \in_R \mathbb{Z}_n$ if A_{m+1} is not among the A_i's for $i = 0, 1, 2, \cdots, m$ and do nothing if A_{m+1} is among the A_i'sas to model the intractability of discrete logarithm. The initial two commands for the oracle would be used to construct the public and private key, so it requires that $A_0 \neq G$ otherwise the private key would be trivial.

Subtract This command takes nothing but return the subtraction $z_{m+1} = z_{m-1} - z_m$ of the two previous exponents pushed. If $A_{m+1} = z_{m+1}G$ is among the elements already stored, return the corresponding exponent. Otherwise randomly select $A_{m+1} \in_R \mathbb{A}_n$.

The Signing Oracle or Hint Command In order to model the signing operation, one assumes that the message signature pair reveals absolute nothing about the underlying private key. So the pairs of message and signature is just random pairs with no correlation. So given message M_{m+1}, the signing oracle randomly pick a group element $A_{m+1} \in_R \mathbb{A}_n$ as a result of imitating a random hash function. Then it randomly picks a exponent $z_{m_1} \in_R \mathbb{Z}_n$ so that in principle no information is revealed from a pair of message and signature signed by the true signer. But if z_{m_1} already exists in the memory (or exists in the previous state) the signing oracle simply returns that exponent to preserve consistency. In the generic group model, hash signature pairs (A_i, z_i) are divided into two groups with the basic ones resulting from a push command and derived ones from elsewhere. By using linear algebra, one can obtain a derived signature from basic ones by solving the corresponding coefficients. But this coincidence happens with negligible probability.

10 Sufficient Security Conditions

Existential Unforgeability against No-Message Attacks Suppose F is an $(\epsilon_F, \tau_F, 0)$-forger in the generic model. The following is the reduction to an inverter or zero finder (I) using F as a subroutine.
Input: e (a randomly selected hash value)

(1) Invoke F with a simulated oracle
(2) Obtain the result from the forger namely the signature (r,s)
(3) Verify the signature using the simulated oracle
Output: a message M such that h(M)=e or h(M)=0.
Since one can build a simulated oracle which is indistinguishable from the generic push and subtract operations, such forger could then be used to build an Inverter or Zero-finder which contradicts the assumption of the hash function. The Theorem says:

Theorem 1 If F is an $(\epsilon_F, \tau_F, 0)$ forger of ECDSA with hash function h and an almost-invertible conversion function f in the generic model for $\mathbb{A}_n$, then there exists an (ϵ_I, τ_I)-inverter and (ϵ_Z, τ_Z)-zero-finder for h with

$$\epsilon_I + \frac{\epsilon_Z}{10\tau_F} \geq \frac{\epsilon_F}{10\tau_F} - \frac{\tau_F}{20n}, \tau_I, \tau_Z \leq 2\tau_F$$

Selective Unforgeability against Adaptive Chosen-Message Attacks Algorithm of reduction to second-preimage-finder from such a forger F
(1). Run the usual oracle for $\mathbb{A}_n$ with some exceptions which the signing oracle might throw.
(2). To answer a signing queries of F during command i for a message M, do it as specified in the previous section. Basically the signing oracle randomly throw some pairs of hash value and signature, which the forger could essentially produces by itself.

Theorem 2 If F is an $(\epsilon_F, \tau_F, q_F)$ forger of ECDSA on any message space $\mathscr{F}$ with hash function h and an almost-invertible conversion function f of strength at least β in the generic model for $\mathbb{A}_n$, then there exists an $(\epsilon_S, \tau_S, \mathscr{F})$-second-preimage-finder S and (ϵ_Z, τ_Z)-zero-finder Z_h where

$$\epsilon_S \geq \epsilon_F - \frac{\tau_F^2}{2n} - (\frac{1}{\tau_F} - \frac{1}{n})^{-1}\beta, \tau_S \leq 2\tau_F$$

Existential Unforgeability Against Adaptive Chosen-Ciphertext Attacks The intuit idea behind this theorem is that one can build a simulated signing oracle which is indistinguishable from a generic model. The simulated oracle use the random hash oracle to generate hash value and the signing oracle mentioned in the building block section to answer the signing queries.

Theorem 3 If there exists an $(\epsilon_F, \tau_F, q_F)$-forger F_h of ECDSA with uniform random hash function h and an almost-invertible conversion function for the generic group model. Then there exists an (ϵ_C, τ_C)-collision-finder C_h and (ϵ_Z, τ_Z)-zero-finder where

$$\epsilon_C + \epsilon_Z \geq \epsilon_F - \frac{\tau_F^2}{2n}, \tau_C, \tau_Z \leq 2\tau_F$$

11 Conclusion

Up till now, the discrete logarithm problem defined on elliptic curve over finite field is the hardest in the sense that the fastest algorithm to solve ECDLP is fully exponential time in contrast to the DLP in a finite field which could be solved in sub-exponential time using Baby-step-giant-step. This method essentially only invokes group operations as a black-box as treated by a generic group. Therefore, basing the signature scheme on such group give some evidence for the assumption of generic group model. Since there have been some weaknesses found in SHA-1 as used in the original paper, SHA-2 would be recommended for practical implementations to account for the assumption of collision-resistance and uniformity which together implies preimage resistance as well. Overall, the ECDSA is a provably secure digital signature scheme assuming (i) uniform and collision-resistant hash function. (ii)pseudorandomness for the generation of ephemeral private keys (iii) generic model of the underlying group on elliptic curve defined over finite field. (iv) some constraints on the mapping from ephemeral public key space to private key space.