Modelling IBE-based Key Exchange Protocol using Tamarin Prover

Srijanee Mookherji, Vanga Odelu and Rajendra Prasath

Indian Institute of Information Technology Sricity, Chittoor, Andhra Pradesh.
{srijanee.mookherji,odelu.vanga,rajendra.prasath}@iiits.in

Abstract. Tamarin Prover is a formal security analysis tool that is used to analyse security properties of various authentication and key exchange protocols. It provides built-ins like Diffie-Hellman, Hashing, XOR, Symmetric and Asymmetric encryption as well as Bilinear pairings. The shortfall in Tamarin Prover is that it does not support elliptic curve point addition operation. In this paper, we present a simple IBE (Identity-Based Encryption) based key exchange protocol and tamarin model. For modelling, we define a function to replace the point addition operation by the concept of pre-computation. We demonstrate that the security model functions for theoretical expectation and is able to resist Man-In-The-Middle (MITM) Attack. This model can be used to analyse the formal security of authentication and key exchange protocols designed based-on the IBE technique.

Keywords: Tamarin Prover, Key Exchange Protocol, Identity Based Encryption (IBE), Man-In-The-Middle (MITM) attack, Elliptic Curve Point Addition Operation

1 Introduction

In the current era of remote interactions among systems, secured communication is of utmost importance. While designing any such communication protocol, one must ensure that a connection is established with authentic users only. There are multiple authentication protocols based on key exchange techniques proposed till date. One such technique is by using the concept of Diffie-Hellman key exchange protocol [DH76]. The authentication protocols use this technique to establish a secret session key among two parties [LLC+08, Pu10]. In case of multi-server environment, the major drawback of such protocols is that the client needs to store the public key of multiple servers. To overcome this issue IBE-based key exchange protocols were introduced. Here the identity of the clients and servers are used as public keys [TL15, ODWC18]. In this paper, we present a generalised key exchange protocol using the concepts of IBE and Diffie Hellman key exchange protocols. We then model it using Tamarin Prover to analyse it's security properties.

Tamarin Prover [MSCB13] is a powerful formal security analysis tool. One can design protocol and specify adversary model to check the security properties of the protocol. Tamarin Prover provides a wide range of built-in features like Diffie-Hellman, hashing, symmetric-encryption, asymmetric-encryption, signing, bilinear-pairing, xor, multiset and revealing-signing. The main Tamarin Prover repository[1] consists of multiple example protocols and attack models. The tool developers have pointed out that Tamarin Prover currently does not support elliptic curve point addition operations [SSCB14]. They provide an example model for 'Identity based authenticated key agreement protocols from pairings' designed by Chen and Kudla [CK03] which is based on Diffie-Hellman key exchange protocol.

The rest of the paper is organized as follows: We first present built-in model simple Diffie Hellman Key Exchange Protocol [MSCB13] and demonstrate that it is vulnerable to MITM attack

[1] https://github.com/tamarin-prover/tamarin-prover

in Section 2. Next in Section 3, we see an overview of the Chen and Kudla Key Agreement protocol [CK03] and study the modelling strategy used in Tamarin Prover to perform Point Addition. The developers discuss the strategy of ordered concatenation of the pairing terms. As it is an Identity based Key Exchange protocol the discussed variant proves to be successful, but this method cannot be used in all protocols using Point Addition. To look into the same, we design a generalised IBE based key exchange protocol which is elaborated in Section 4. To model the protocol we required point addition operation support. As Tamarin Prover does not support the same as well as equalities like $(c)[(a)P + (b)]P = [(ca)P + (cb)P]$ [SSCB14], we normalise our model by defining a function $hf/1$ which returns a single output against multiple input, working similar to that of a hash function. Along with this some pre-computations need to be done in order to model our protocol. The same is described in Section 4.3.1. We show in Section 4.4 that our model functions fine and that the designed key exchange protocol resists MITM attack. We then concluded the paper in Section 5.

2 Diffie Hellman Key Exchange Protocol

In this section, we present the review of the Diffie-Hellman protocol and it's modeling in Tamarin Prover.

2.1 Simple Diffie Hellman Key Exchange Protocol

Diffie Hellman key exchange protocol [DH76] is used to establish a shared secret over a public channel. The protocol is vulnerable to MITM attack. An adversary is able to pretend as one of the parties and establish a shared secret key with the other party [JG02]. Table 1 shows the summary of the Diffie-Hellman key exchange protocol.

Table 1: Diffie Hellman Key Exchange Protocol

Alice	Bob
Chooses secret a	Chooses secret b
Computes $Y = g^a$	Computes $X = g^b$
$\xrightarrow{Y=g^a}$	
	$\xleftarrow{X=g^b}$
Compute $A = X^a$	Compute $B = Y^b$
Alice and Bob Establish shared secret key $A = B = g^{ab}$	

2.2 Modelling using Tamarin Prover

The Tamarin security model of the Diffie Hellman Key Exchange Protocol is as follows.

```
theory diffieplain
begin

builtins: diffie-hellman

/* protocol */
rule Init_client:
   [ Fr(~a)]
   -->
   [ Keys(~a, 'g'^~a),Out('g'^~a)]
```

```
rule Init_server:
   [ Fr(~b) ]
   -->
   [ Keys(~b, 'g'^~b),Out('g'^~b)]

rule Init_Finish_server:
  let key_other = 'g'^a
  in
   [ Keys(b, 'g'^b), In( 'g'^a ) ]
   --[ Session_created(key_other^b) ]->
   [ Session(key_other^b)]

rule Finish_client:
  let key_other = 'g'^b
  in
   [ Keys(a, 'g'^a), In( 'g'^b ) ]
   --[ Session_created(key_other^a) ]->
   [ Session(key_other^a) ]

/* Key Reveals */
rule key_reveal:
   [ Keys(~secretk, 'g'^~secretk) ]
   --[ Key_reveal(~secretk) ]->
   [ Out(~secretk) ]

rule Session_reveal:
  [ Session(skey) ]
  --[ Reveal_sessionKey(skey)]->
  [ Out(skey)]
```

The lemma used in the model 2.2 is a simple Man-In-The-Middle attack. The code is stated below. The description of the lemma is provided in Lemma 1.

```
// lemma
lemma MITM:

  "(All #i1 skey .
    (
      Session_created(skey) @ i1
      &
      not ( (Ex A #ia .
        Key_reveal( A ) @ ia)
        | (Ex B #ib . Reveal_sessionKey
        ( B ) @ ib )
        )
    )
    ==> not (Ex #i2. K( skey ) @ i2 )
  )"
```

Lemma 1. *For all the cases where the Secret Session keys are created and, neither the secret keys of both the parties nor is the shared secret key is revealed, the Adversary should not be able to compute the Secret Session Key.*

2.3 Model Visualisation

The visualization display of the Tamarin Prover shows that MITM attack is possible in a Simple Diffie Hellman Key Exchange protocol. The Figure 1 demonstrates "trace found" and the lemma has turned red while the lemma was solved. It signifies that the tool was able to find a case where the Adversary can compute the Secret Session Key. The Figure 2 shows a graphical representation of the same. The red line indicates the value that the Adversary is able to compute and perform the MITM attack.

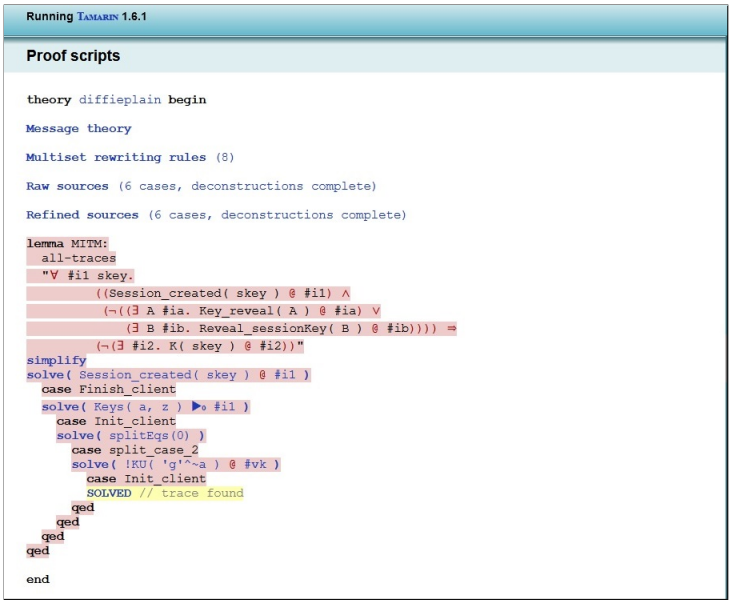

Figure 1: Tamarin Prover Model Visualisation - Lemma

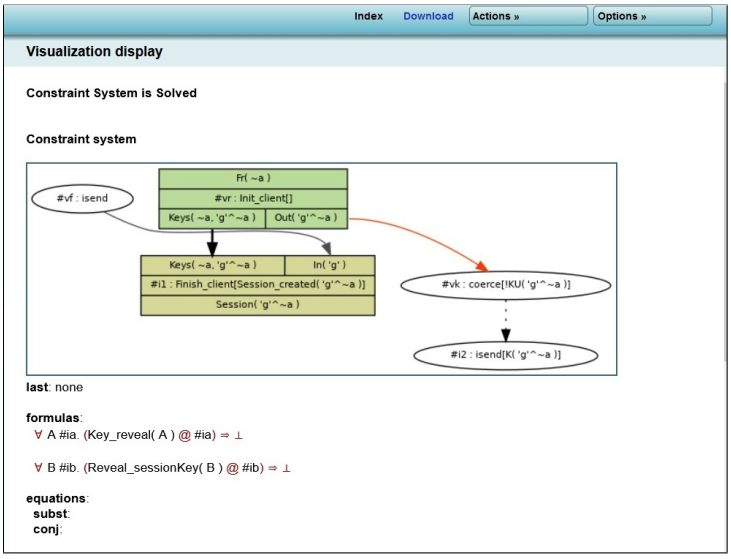

Figure 2: Tamarin Prover Model Visualisation - Graph

3 Study of Tamarin Prover Modelling of Chen and Kudla Protocol

The Chen and Kudla protocol[CK03] is summarised in the Table 2 and 3. In the KGC Setup phase (Table 2), the KGC chooses private key,s and computes public key $P_s = sP$. Then 2 users Alice, A and Bob, B is considered. KGC computes Compute $Q_A = h(ID_A)$, $S_A = sQ_A$ and sends S_A to Alice and S_B to Bob. In the Key Exchange Phase (Table 3) A and B compute $T_A = aP$ and $T_B = bP$ where a and b are chosen secrets. A sends T_A to B and B sends T_B to A. Finally, Alice and Bob computes $K_{AB} = \hat{e}(S_A, T_B)\hat{e}(aQ_B, P_s)$ and $K_{BA} = \hat{e}(S_B, T_A)\hat{e}(bQ_A, P_s)$ which results in the computation of $K = K_{AB} = K_{BA} = \hat{e}(bQ_A + aQ_B, P_s)$.

Table 2: Chen Kudla Protocol- KGC Setup and User Registration Phase

Alice	**KGC**	**Bob**
	KGC Setup	
	Chooses private key, s Generates public key $P_s = sP$	
	User Registration	
	Compute $Q_A = h(ID_A)$, $S_A = sQ_A$	
	$\xleftarrow{TA \rightarrow Alice: <S_A>}$	
	Compute $Q_B = h(ID_B)$, $S_B = sQ_B$	
	$\xrightarrow{TA \rightarrow Bob: <S_B>}$	

Table 3: Chen Kudla Protocol- Key Exchange Phase

Alice	**Bob**
Choose secret a	Choose secret b
Computes $T_A = aP$	Computes $T_B = bP$
$\xrightarrow{ToBob: <T_A>}$	
	$\xleftarrow{ToAlice: <T_B>}$
Compute $K_{AB} = \hat{e}(S_A, T_B)\hat{e}(aQ_B, P_s)$	
	Compute $K_{BA} = \hat{e}(S_B, T_A)\hat{e}(bQ_A, P_s)$
Shared Secret $K = K_{AB} = K_{BA} = \hat{e}(bQ_A + aQ_B, P_s)$	

In the model example $Chen\_Kudla.spthy$ present in the repository[2], the multiplication of the bilinear terms are replaced by the concatenation function [SSCB14]. The model functions appropriately. The code snippet is as follows:

```
rule Init_2:
  let skA = pmult(~s1, hp($A))
```

[2]https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/ake/bilinear

```
      mpk = pmult(~s2,'P')
      X   = pmult(~ex,'P')
      sessKey = kdf( em(hp($B), mpk)^~ex,
      em(skA, Y), pmult(~ex,Y), $A, $B, X, Y )

rule Resp_1:
  let skB = pmult(~msk, hp($B))
      mpk = pmult(~msk, 'P')
      Y = pmult(~ey,'P')
      sessKey = kdf( em(skB, X),
      em(hp($A), mpk)^~ey,
      pmult(~ey,X), $A, $B, X, Y)
```

4 IBE based Key Exchange Protocol

As discussed in Section 2, Diffie Hellman Key Exchange Protocol suffers from MITM attack. To secure the protocol we design an IBE based key exchange protocol using the concept of Diffie Hellman Key Exchange Protocol.

4.1 Mathematical Preliminaries

4.1.1 Bilinear Pairings

Bilinear Pairings can be defined as follows. Assume G_1 is an additive cyclic group of prime order q, G_2 is a multiplicative cyclic group of prime order q. P be the generator of G_1. The bilinear pairing equation $\hat{e} : G_1 \times G_1 \rightarrow G_2$ satisfies the following properties [TL15].

- Bilinearity: $\hat{e}(xP, yQ) = \hat{e}(P, Q)^{xy}$, for all $P, Q \in G1$ and for all$x, y \in Z_q^*$
- Computability: For all $P, Q \in G1$, $\hat{e}(P, Q)$ can be computed efficiently.
- Non-degeneracy: There exist $P, Q \in G_1$ with $\hat{e}(P, Q) \neq 1$, where 1 is the multiplicative identity of G_2.

Definition 1. Collusion Attack Algorithm With k-Traitors (k-CAA Problem)
Given $P \in G1, sP \in G1, x_1, x_2, ..., x_k \in Z_q^*$ and $(\frac{1}{s+x_1})P, (\frac{1}{s+x_2})P, ..., (\frac{1}{s+x_k})P$ for an integer $k, s \in Z_q^*, P \in G_1$, it is hard to compute $(s + x)^{-1}P$, where $x \notin x_1, x_2, ..., x_k$.

4.2 Key Exchange Protocol

Table 4 and Table 5 describes the protocol that we have designed. Key Generation Center (KGC) setup phase begins with the KGC choosing a master private key, K and generating it's own public key $Q = KP$.

In the registration phase, we assume that a single Server,S and a single Client,C registers with the KGC. The KGC provides $< ID_C, K_C >$ to C and $< ID_S, K_S >$ to S. Here, $K_C = \frac{1}{h(ID_C)+K}P$ and $K_S = \frac{1}{h(ID_S)+K}P$.

In the Key Exchange Phase Client,C chooses secret x and computes $A = x(h(ID_S)P + Q)$. Similarly, Server,S chooses secret y and computes $B = x(h(ID_C)P+Q)$. Client then sends A to Server and Server sends B to Client. On receiving the same Client further computes $SK = \hat{e}(B, K_C)^x$ and Server computes $SK = \hat{e}(A, K_S)^y$.

Table 4: Client and Server Registration with KGC

Client	KGC	Server
	KGC Setup Phase Chooses Secret Key, K Computes Public Key, Q = KP	
	Client Registration Phase $\xleftarrow{KGC \rightarrow Client: <ID_C, K_C>}$ $K_C = \frac{1}{h(ID_C)+K} P$	
	Server Registration Phase $\xrightarrow{KGC \rightarrow Server: <ID_S, K_S>}$ $K_S = \frac{1}{h(ID_S)+K} P$	

Table 5: Client-Server Key Exchange Protocol

Client	Server
Choose secret x $A = x(h(ID_S)P + Q)$ $\xrightarrow{<A>}$	
	Choose secret y $B = y(h(ID_C)P + Q)$ $\xleftarrow{<B>}$
Compute : $SK = \hat{e}(B, K_C)^x$	Compute : $SK = \hat{e}(A, K_S)^y$

4.3 Modelling using Tamarin Prover

4.3.1 Normalization and Precomputation

As per the discussion in Section 1 point addition operation is not supported in Tamarin Prover. The Chen and Kudla protocol [CK03] is an Identity based key exchange protocol with point addition being a part of the bilinear terms thus, concatenation was a solution. In the designed IBE based key exchange protocol the point addition needs to be normalised and precomputed. Hence, we define a function $hf/1$ that provides a single output against multiple inputs. We normalise our protocol in the following way :

- $K_C = \frac{1}{h(ID_C)+K} P$ is normalised as $hf(ID_C, K)$. The further computation is continued as $K_C = pmult(inv(hf(ID_C, K)),' P')$. Here inv denotes inverse and $pmult$ denotes point multiplication.
- $K_S = \frac{1}{h(ID_S)+K} P$ is normalised as $hf(ID_S, K)$. The further computation is continued as $K_C = pmult(inv(hf(ID_S, K)),' P')$. Here inv denotes inverse and $pmult$ denotes point

multiplication.

In order to use the above normalisation, in the key exchange phase $A = x(h(ID_S)P + Q)$ and $B = y(h(ID_C)P + Q)$ cannot be directly computed. We precompute the values K_C and K_S as mentioned above, at KGC and send them to the Client and Server.

4.3.2 Tamarin Prover Code

We model our IBE based Key Exchange protocol in Tamarin Prover.

```
theory test
begin
builtins: diffie-hellman,
bilinear-pairing, hashing

functions: hf/1

rule Kgc:

let
Q = pmult(~Key,'P')

in
[Fr(~Key)]
--[KGCSetup(),OnlyOnce()]->
[!Ltk($KGC,~Key),!Pkk($KGC,Q)]

rule CReg:

let
TempKc = hf(h($C),Key)
Kc = pmult(inv(TempKc),'P')
in
[!Ltk($KGC,Key)]
--[OnlyOnce()]->
[!PKc($C,TempKc),!Ltk($C,Kc)]

rule SReg:
let
TempKs = hf(h($S),Key)
Ks = pmult(inv(TempKs),'P')
in
[!Ltk($KGC,Key)]
--[OnlyOnce()]->
[!PKs($S,TempKs),!Ltk($S,Ks)]


rule Client:

let
xp = pmult(TempKs,'P')
A = pmult(~x,xp)
in
[!PKs($S,TempKs),Fr(~x)]
```

```
--[]->
[Out(<$S,A>),Client(~x),Clientid(A)]


rule Server:

let
yp = pmult(TempKc,'P')
B = pmult(~y,yp)
in
[!PKc($C,TempKc),Fr(~y)]
--[]->
[Out(<$C,B>),Server(~y),Serverid(B)]

rule ClientSession:

let
sk = em(B,Kc)
sky = sk^~y
in
[Server(~y),!Ltk($C,Kc),In(<$C,B>),
Client(~x),Clientid(A)]
--[Session_created_C(sky),
Accept( ~y, $C, $S, sky )
, Sid( ~y, <'Client',$C,$S,A,B> )
, Match( ~y, <'Server',$S,$C,A,B> )]->
[Session(sky)]

rule ServerSession:

let
sk = em(A,Ks)
skx = sk^~x
in
[Client(~x),!Ltk($S,Ks),In(<$S,A>),
Server(~y),Serverid(B)]
--[Session_created_S(skx),
Accept( ~x, $S, $C, skx )
, Sid( ~x, <'Server',$S,$C,A,B> )
, Match( ~y, <'Client',$C,$S,A,B> )]->
[Session(skx)]

restriction OnlyOnce:
"All #i #j. OnlyOnce()@#i
& OnlyOnce()@#j ==> #i = #j"

/* Key Reveals */
rule ltk_reveal:
   [ !Ltk($KGC, ~Key) ]
   --[ LtkRev($KGC) ]->
   [ Out(~Key) ]
```

```
rule Session_reveal:
  [ Session(skey) ]
  --[ SesskRev(skey)]->
  [ Out(skey)]
```

We use the same lemma, Lemma 1 which was used to model MITM attack on a simple Diffie Hellman protocol in Section 2.

4.4 Model Visualisation

We can see in Fig. 3. that after solving the lemma colour has turned green. It means no traces were found of any Adversary able to compute the Secret Session Key. Thus, we can safely say that the designed IBE based key exchange protocol resists MITM attack.

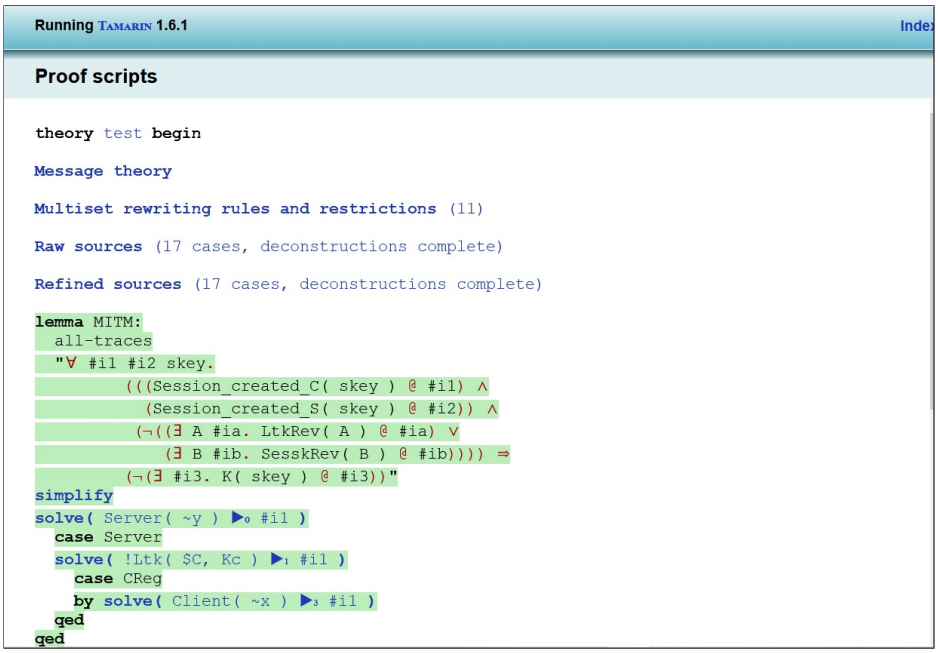

Figure 3: Tamarin Prover Model Visualisation for IBE based Key Exchange Protocol

5 Conclusion and Future Work

In this paper we first modelled a simple Diffie Hellman key exchange protocol and using Tamarin Prover we see that it is susceptible to MITM attack. Next we design a simple generalised IBE based key exchange protocol using the concept of Diffie Hellman Key Exchange protocol. We model the same using Tamarin Prover and use hf/1 function for modelling a point addition operation used in the protocol. We finally prove that the model works fine and is secured against MITM attack. Thus, IBE based key exchange and authentication protocols can be designed using the modelling strategy discussed in this paper.

In the future, we plan to use the same modelling concept to design an IBE based key exchange and authentication protocol.

References

[CK03] Liqun Chen and Caroline Kudla. Identity based authenticated key agreement protocols from pairings. In *16th IEEE Computer Security Foundations Workshop, 2003. Proceedings.*, pages 219–233. IEEE, 2003.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[JG02] Anna M Johnston and Peter S Gemmell. Authenticated key exchange provably secure against the man-in-the-middle attack. *Journal of cryptology*, 15(2):139–148, 2002.

[LLC+08] Rongxing Lu, Xiaodong Lin, Zhenfu Cao, Liuquan Qin, and Xiaohui Liang. A simple deniable authentication protocol based on the diffie–hellman algorithm. *International Journal of Computer Mathematics*, 85(9):1315–1323, 2008.

[MSCB13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer, 2013.

[ODWC18] Vanga Odelu, Ashok Kumar Das, Mohammad Wazid, and Mauro Conti. Provably secure authenticated key agreement scheme for smart grid. *IEEE Transactions on Smart Grid*, 9(3):1900–1910, 2018.

[Pu10] Qiong Pu. An improved two-factor authentication protocol. In *2010 Second International Conference on Multimedia and Information Technology*, volume 2, pages 223–226, 2010.

[SSCB14] Benedikt Schmidt, Ralf Sasse, Cas Cremers, and David Basin. Automated verification of group key agreement protocols. In *2014 IEEE Symposium on Security and Privacy*, pages 179–194. IEEE, 2014.

[TL15] Jia-Lun Tsai and Nai-Wei Lo. Secure anonymous key distribution scheme for smart grid. *IEEE transactions on smart grid*, 7(2):906–914, 2015.