Improving a privacy-preserving clone detection mechanism with elliptic curve cryptography

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master Thesis

Improving a Privacy-preserving Clone Detection Mechanism with Elliptic Curve Cryptography

Lars Krapf

15. September 2010

Advisors: Prof. Dr. Srdjan Čapkun, Davide Zanetti
Department of Computer Science, ETH Zürich

In this thesis we improve an existing protocol for privacy-preserving counterfeit detection with elliptic curve cryptography. While providing equivalent security versus passive adversaries, the new protocol is superior in terms of execution time due to the smaller key lengths necessary for cryptography. The protocol also proves to be less sensitive to external factors like network latency, compared to implementations in general MPC frameworks, because of its peer-to-peer architecture.

Acknowledgments

I would like to thank my supervisor Davide Zanetti for his valuable guidance and support during the whole development of this thesis. Further I would like to dedicate this work to my mother Caterina Venturi for her continuous support during my entire studies. Without her nothing of this would ever have been possible.

Lars Krapf

CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

In this chapter we introduce the problem of detecting counterfeit products in RFID-enabled supply chains in a privacy-preserving way.

1.1 The clone detection problem

When counterfeit products are injected into legal supply chains, for example when a distributor unknowingly purchases seemingly genuine products from an illegitimate source, several problems arise: Through sales and reputation losses considerable damage is done to the producers of the genuine product (OECD estimates 200 billion US dollars of losses through counterfeit products in 2005 [16]). Moreover, fabricated products, like for example counterfeit pharmaceuticals, pose significant health risks to consumers.

1.1.1 RFID-enabled supply chains

A supply chain is the system of all partners and operations involved in the logistics of a product, from the producer, through all distributors up to the retailer. To uniquely identify products along their way to the consumer, products can be equipped with radio frequency identification (RFID) tags, which have several advantages over traditional identification methods like barcodes. The partners along the way can read the identifiers, and store or process information associated with them. We call such a system an RFID-enabled supply chain.

1.1.2 Injection of counterfeits

In order to inject counterfeit products into RFID-enabled supply chains, the forgers have to create identical copies of the corresponding RFID-tags also, so-called *clones*. Because manual product inspection is cumbersome and often difficult, finding an automated

solution based on the existing RFID infrastructure is eligible. We call this problem *clone detection in RFID-enabled supply chains.*

1.1.3 Privacy Preservation

In a competitive market environment, players are often reluctant to share information about their trade volume, business partners or product flows with other competitors. Thus it is required to detect counterfeits in a privacy-preserving way, meaning that as little information as possible is shared between the partners involved in such a computation.

1.2 An existing solution

In [48] Davide Zanetti has presented an appealing trace-based solution to the clone-detection problem based on a simple observation: In a legitimate supply chain every recording of a product entering at a distributors location (RCV) must be followed by a recording of the same product leaving this location (SHP). Conversely a recording of a product entering a location must have been preceded by a recording of the same product leaving at another location. This observation is the basis for the protocols discussed below.

1.2.1 Model

Whenever a product is shipped to or received by a partner its RFID tag gets scanned and the corresponding event is stored into a (private) database. Throughout the rest of this text such an event is represented by a quadruple $e = (ID, T, L, D)$, where ID is the unique identifier of the product, T and L denote the time and location this event was recorded, and D is the type of the activity performed (i.e. shipping or receiving).

1.2.2 Algorithm

The introductory observation can be formalized as follows: For every time-consecutive pair of events (e_1, e_2) exactly one of the following propositions must hold

$$R_1 : [e_1(L) = e_2(L)] \wedge [e_1(D) = RCV] \wedge [e_2(D) = SHP]$$
$$R_2 : [e_1(L) \neq e_2(L)] \wedge [e_1(D) = SHP] \wedge [e_2(D) = RCV]$$

Although the verification of these rules is sufficient for clone-detection, in [22] a more robust algorithm was given, which will be called BT throughout this text.
Due to tag malfunctions and misreads erroneous event records may be present even if no clone injection has been performed. We therefore model reads as independent binomially-distributed random variables, with misread probability p_{mr}. We can then evaluate the presence of clones based on a hypothesis test with significance level α. The

L_i	S_i	L_j	S_j	Failed Rule	Missing events (N^u_{ME})
A	RCV	A	RCV	R_1	A-SHP, B-RCV, B-SHP (3)
A	RCV	B	RCV	R_1	A-SHP (1)
A	RCV	B	SHP	R_1	A-SHP, B-RCV (2)
A	SHP	A	RCV	R_2	B-RCV, B-SHP (2)
A	SHP	A	SHP	R_2	B-RCV, B-SHP, A-RCV (3)
A	SHP	B	SHP	R_2	B-RCV (1)

Table 1.1: Relation between failed rule verification and missing events

null-hypothesis H_0 states that failures are caused by misreads, whereas the alternate hypothesis H_A implies the presence of a cloned product.

$$Pr(K \geq k) = \sum_{k=N_{ME}}^{N_{TE}} \binom{N_{TE}}{k} {p_{mr}}^k (1 - p_{mr})^{N_{TE}-k} \tag{1.1}$$

$$Pr(K \geq k) \leq \alpha \rightarrow H_0 \tag{1.2}$$

$$Pr(K \geq k) > \alpha \rightarrow H_A \tag{1.3}$$

We denote the total number of considered events with N_{TE} and the total number of missing events as N_{ME}.
To get N_{ME} we sum up the number of missing events for every time-consecutive pair (e_i, e_j) for which both of the above stated propositions R_1 and R_2 do not hold, i.e the number of events that would need to be inserted between e_i and e_j to pass rule verification, as exemplified in table 1.1.

The total number of missing events is therefore given by

$$N_{ME} = \sum_u N^u_{ME} \tag{1.4}$$

for each failed rule verification u.

1.2.3 Discussion

As we have seen, to compute the total number of missing events we need to sort the events according to their respective time indices. For each pair of time-consecutive events we then evaluate R_1 and R_2 and use equation 1.4 to sum up the total number of missing events N_{ME}. The challenge lies in efficiently performing this algorithm in a privacy-preserving way, i.e. with no partner learning anything about the other partners events. To achieve this we use the concepts of secure multiparty computation (SMC).
In [14] the BT algorithm has been implemented in different existing SMC frameworks,

and an ad-hoc protocol was presented that allows the partners to compute the number of missing events in a peer-to-peer fashion. Although demonstrating promising results in terms of asymptotic complexity, the performance of the latter approach is very dependent on the keysize of the underlying cryptographic scheme due to the large number of encryptions necessary to evaluate Yao circuits, the core component of the presented protocol.

1.3 A possible Improvement

Elliptic curve cryptography, i.e. cryptography based on the discrete logarithm problem on the group of elliptic curve points (ECDLP), has caught quite a lot of attention over the past two decades. Because there is no known algorithm to solve the ECDLP in subexponential time, it is believed that elliptic curve cryptography can provide security equivalent to that of other public key cryptosystems (i.e. RSA), but with much smaller key sizes. This makes cryptographic schemes based on elliptic curves an interesting choice in cases where computing-power is limited, as it is the case for smartcards or sensor networks, or when an efficient encryption operation is necessary as a building block for a larger cryptographic scheme.
Since keylength is a critical factor for the performance of the ad-hoc protocol, the use of elliptic curve cryptography was suggested as a possible optimization. The design and implementation of a protocol improved in such a way has been the main focus of this work.

CHAPTER 2

Problem Statement

In this chapter we review the work so far and discuss the previous implementation of the BT algorithm. We show limitations of the existing implementation and propose a solution.

2.1 Previous work

2.1.1 VIFF implementation

The Virtual Ideal Functionality Framework (VIFF) is a secure multiparty computation framework written in Python [5]. It allows secure evaluation of relations and arithmetic operations over shared values, based on the concepts of pseudo-random and Shamir secret sharings. It provides different runtimes that allow for secure computation versus passive or active adversaries respectively.
In [14] a VIFF implementation of the *BT* algorithm has been given which serves as a basis for comparison for our peer-to-peer approach.

2.1.2 Ad-hoc protocol and Keysize

In [14] Leo Fellmann also designed and implemented an ad-hoc protocol (PP) that allows the partners of an RFID-enabled supply chain to cooperatively compute the *BT* predicate presented in section 1.2.2 in a privacy preserving way and without an intermediate trusted party.
Although his peer-to-peer implementation proved to have advantages over traditional secret sharing based approaches, the high number of encryption operations (especially in the subprotocol for oblivious transfer) made the size of the cryptographic keys a critical performance factor. Because the computation of the protocol proved infeasible for larger keys and problem instances, the application of elliptic curve cryptography was suggested as a possible optimization.

2.1.3 Missing evaluations

The experimental evaluations of the ad-hoc protocol were partially inaccurate [1] , especially for larger keysizes, as well as missing several important test-cases.

2.2 Contributions

2.2.1 Survey on Elliptic Curve Cryptography

After a short introduction to elliptic curve cryptography in general we give a comprehensive overview over the research in the area. We analyze different parameters of such a system like the representation and algorithms of the underlying finite field, as well as formulae for performing arithmetics on elliptic curve points using several different coordinate systems and curve representations.
We compare the different methods in terms of complexity, and motivate the choices we made for our protocol.

2.2.2 Design and implementation of an improved protocol

We established an improved protocol by applying elliptic curve cryptography to the existing solution and its subprotocols. We explain the cryptographic primitives that are used and discuss all the necessary modifications to the existing protocol. We then present the enhanced ECPP protocol and an implementation thereof.

2.2.3 Evaluation

Finally we do experimental evaluations of several aspects of the new protocol. We analyze the impact of keysize, network latency and CPU handicap for different numbers of partners and events, and compare it to the previous implementations. Additionally we provide a performance analysis of the VIFF implementation versus active adversaries.

[1] This was caused by instabilities in the implementation of the underlying network IO operations

CHAPTER 3

Elliptic Curve Cryptography

In this chapter we provide an introduction to elliptic curve cryptography and its application to our problem. We explain the basic building blocks and parameters of an elliptic curve cryptosystem and analyze its security properties. Then we give an overview of the existing research.

3.1 Elliptic Curves over finite Fields

3.1.1 Definition

An elliptic curve $\mathbf{E}_k$ defined over a field $\mathbb{F}_q$ of characteristic $\neq 2$ or 3 is the set of solutions $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ to the equation

$$y^2 = x^3 + ax + b, \quad a, b \in \mathbb{F}_q \tag{3.1}$$

If $\mathbb{F}_q$ is a field of characteristic 2 (which is of special interest regarding efficient implementation in hard- and software) then the resulting formula can be reduced to the form

$$y^2 + cxy + dy = x^3 + ax + b, \quad a, b, c, d \in \mathbb{F}_q \tag{3.2}$$

3.1.2 The Group Law

The straight line joining two points P and Q on $\mathbf{E}$ must intersect the curve at another point R. If we then reflect this point in the x-axis we obtain another point which we will call $P + Q$. To add P to itself, we take the tangent to the curve at P and again reflect its intersection point with $\mathbf{E}$ in the x-axis to obtain the point $P + P$. If the tangent is vertical, it 'intersects' the curve at the point at infinity and $P + P = \mathcal{O}$.

Together with this special point $\mathcal{O}$, the points on an elliptic curve $\mathbf{E}$ over $\mathbb{F}_q$ form an (additive) abelian group with $\mathcal{O}$ serving as its identity element. We now give formulae for the addition of two curve points represented in affine coordinates.

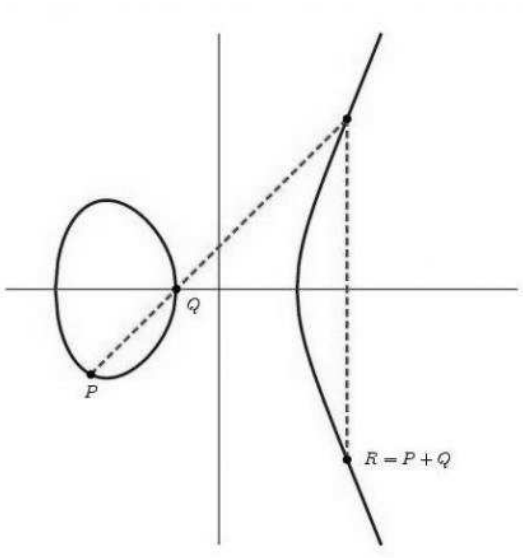

Figure 3.1: Addition of two curve points

For $\mathbf{E}(\mathbb{F}_q)$, with $\mathbb{F}_q$ of characteristic > 3 the formulae for the group law are

$$-P = (x_1, -y_1)$$

When $x_1 \neq x_2$ we set

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

and when $x_1 = x_2, y_1 \neq 0$ we set

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

If

$$P + Q = (x_3, y_3) \neq \mathcal{O}$$

then

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

For (ordinary) $\mathbf{E}(\mathbb{F}_q)$ with $\mathbb{F}_q = \mathbb{F}_{2^p}$ the formulae are

$$-P = (x_1, y_1 + x_1)$$

When $x_1 \neq x_2$ we set

$$\lambda = \frac{y_2 + y_1}{x_2 + x_1},\ \mu = \frac{y_1 x_2 + y_2 x_1}{x_2 + x_1}$$

and when $x_1 = x_2 \neq 0$ we set

$$\lambda = \frac{x_1^2 + y_1}{x_1},\ \mu = x_1^2$$

If

$$P + Q = (x_3, y_3) \neq \mathcal{O}$$

then

$$x_3 = \lambda^2 + \lambda + a_2 + x_1 + x_2$$

$$y_3 = (\lambda + 1)x_3 + \mu = (x_1 + x_3)\lambda + x_3 + y_1$$

A more explicit discussion of these formulae is given in [25].

3.1.3 Underlying Field

Although elliptic curves can be defined over numerous domains, for cryptographic usage two types of underlying fields are predominant: Curves defined over finite fields of prime order ($\mathbb{F}_p$), and curves defined over finite fields of characteristic two ($\mathbb{F}_{2^m}$).
In the former case field elements are often represented by integers and the well-known algorithms for finite field arithmetic apply. In the latter case two different representations have commonly been used: We can either represent field elements by binary polynomials $f(X) = \sum_i^{d-1} x_i X^i, x_i \in \mathbb{F}_2$ modulo an irreducible polynomial of degree d, or use a normal basis $(\alpha, \alpha^2, \ldots, \alpha^{2^{d-1}})$.
Because reduction polynomials with a small number of non-zero coefficients (i.e. trinomials or pentanomials) can be found for most d, there exist very efficient reduction algorithms for field elements represented by binary polynomials, whereas in a normal-basis representation squarings can be efficiently implemented using a simple cyclic shift of the coordinates. Depending on the hardware and runtime environment either one of those alternatives can be advantageous. For the rest of this text we assume field elements are given in polynomial representation.

3.1.4 Coordinate Systems

An elliptic curve point can be represented by several different coordinate systems. For each such system the formulae for addition and point-doubling are different. If projective coordinates are used, these operations can be computed without performing inversions on the elements of the underlying field at the cost of additional squarings and multiplications.
Because on todays hardware the inversion-to-multiplication-ratio in terms of time complexity is usually large, regardless of the chosen finite field or field element representation, the choice of an appropriate coordinate system is of utmost importance to the performance of the implementation.

3.2 ElGamal over Elliptic Curves

As suggested independently by Koblitz [29] and Miller [36], the abelian group of curve points on $\mathbf{E}(\mathbb{F}_q)$ can be used to implement asymmetric cryptographic schemes based on the elliptic curve discrete logarithm problem (ECDLP), (that means finding k for a given a curve point $k \cdot G$ is hard), i.e. the ElGamal public-key cryptosystem.

Assume an elliptic curve $\mathbf{E}$ over $\mathbb{F}_q$ of order $\#\mathbf{E}$ with a fixed point G (preferably a generator of $\mathbf{E}$). Alice wants to send an encoded message $P_m \in \mathbf{E}$ to Bob. Bob publishes his public key consisting of $(G, H = xG)$ where x is his private key.

1. Alice generates a random integer $k \in \{1, \ldots, (\#\mathbf{E} - 1)\}$ and computes
$$A = kG, B = P_m + kH$$
2. Alice sends (A, B) to Bob.
3. Bob can decrypt the message using the equation
$$B - xA = (P_m + kH) - xkG = P_m + xkG - xkG = P_m$$

3.2.1 Security

Today the best known algorithms for solving the discrete logarithm problem (DLP) in $\mathbb{F}_p^*$ are only of sub-exponential complexity in the size $N = \lceil log_2 p \rceil$ of the field elements [6], whereas for solving it on a well-chosen curve $\mathbf{E}(\mathbb{F}_q)$ the best method is of complexity exponential in $n = \lceil log_2 q \rceil$. There are instances of the problem for which more efficient algorithms have been found like the MOV Attack [34] or Pollards method of tame and wild kangaroos [43]. Nevertheless it should be noted that these methods will not be effective if the curve parameters are carefully chosen and then only the general exponential-time methods apply. Hence using a cryptosystem based on elliptic curves, one can provide security equivalent to that of other public-key schemes, but with significantly smaller key sizes.
From on the above asymptotic complexities a formula was derived in [25] that allows for a comparison of key lengths providing roughly equivalent security in both cases

$$n = \beta N^{1/3} (log(N log 2))^{2/3} \tag{3.3}$$

with $\beta \approx 4.91$. Now, n and N can be interpreted as the key lengths in bits of the corresponding cryptosystems. See Figure 3.2.

3.3 Application to the PP protocol

The ElGamal encryption scheme is the main building block employed throughout all stages of the PP protocol, especially in the subprotocol for oblivious transfer used to evaluate the Yao circuits. It was previously implemented using modular exponentiations in $\mathbb{F}_p$, where the length of the cryptographic keys was identified as a critical performance

factor.
Elliptic curve equivalents exist for all cryptographic primitives used in the construction of the protocol. It is thus possible to construct a protocol for anti-counterfeiting providing the same security as PP but using much smaller keylengths.

3.4 Existing Research

Put aside introductory and survey papers like [28], [30], [29], [36] or [44] the research on the use of elliptic curves for cryptography roughly divides into three parts: The analysis of different coordinate systems and resulting formulae for point arithmetic, the construction of algorithms for point multiplication and implementations of cryptographic schemes on different hard- and software architectures.

3.4.1 Coordinate Systems

As we have seen, the choice of the coordinate system has a significant impact on the performance of the addition and doubling algorithms, and thus is of central importance to the design of an elliptic curve implementation. In addition to affine coordinates ($\mathcal{A}$) in [31] projective coordinates ($\mathcal{P}$) were introduced. Other representations include Jacobian ($\mathcal{J}$) [38], Chudnovsky-Jacobian ($\mathcal{J}^c$) [10], and Lopez-Dahab ($\mathcal{LD}$) [33] coordinates.
In the seminal paper [20] the concept of mixed coordinate systems was introduced. The basic idea is to use a different coordinate system for the distinct stages of a scalar multiplication, thus reducing the necessary number of operations in the underlying field. In table 3.4.1 we compare the performance of addition and doubling formulae for different coordinate systems in terms of necessary field operations for curves defined over F_{2^m}. We denote the cost of a field multiplication, inversion or squaring with M, I and S respectively, whereas M_2 denotes multiplications with the curve-parameter a, which can be chosen arbitrarily (without loss of security) [1].

3.4.2 Multiplication Algorithms

A significant part of the existing research is devoted to finding efficient algorithms to perform scalar multiplication of elliptic curve points. Basic algorithms based on Montgomery's method can be found in [33] and [32]. These can be further optimized by using sliding-window methods as in [18]. In [27] the doubling operations in the double-and-add algorithm are replaced by an operation called point-halving. This method is refined in [15]. For curves that have efficiently-computable endomorphisms there exists a very efficient algorithm given in [17].
By rewriting the addition formula one can reduce the number of necessary field multiplications as shown in [7]. A special mention is due to the more recent work of Harold Edwards [13]. His comprehensive study of a new normal form of elliptic curves lead to new insights and efficient arithmetic formulae.

[1] The NIST curves used for our implementation all have $a = 1$ which saves us one field multiplication

Doubling		Addition	
Operation	Costs	Operation	Costs
$2\mathcal{P}$	$7M + 4S + M_2$	$\mathcal{J} + \mathcal{J}$	$15M + 3S + M_2$
$2\mathcal{J}$	$5M + 5S$	$\mathcal{P} + \mathcal{P}$	$15M + 2S + M_2$
$2\mathcal{LD}$	$4M + 4S + M_2$	$\mathcal{LD} + \mathcal{LD}$	$13M + 4S$
$2\mathcal{A} = \mathcal{P}$	$5M + 2S + M_2$	$\mathcal{P} + \mathcal{A} = \mathcal{P}$	$11M + 2S + M_2$
$2\mathcal{A} = \mathcal{LD}$	$2M + 3S + M_2$	$\mathcal{J} + \mathcal{A} = \mathcal{J}$	$10M + 3S + M_2$
$2\mathcal{A} = \mathcal{J}$	$M + 2S + M_2$	$\mathcal{LD} + \mathcal{A} = \mathcal{LD}$	$8M + 5S + M_2$
-	-	$\mathcal{A} + \mathcal{A} = \mathcal{LD}$	$5M + 2S + M_2$
-	-	$\mathcal{A} + \mathcal{A} = \mathcal{J}$	$4M + S + M_2$
$2\mathcal{A}$	$I + 2M + S$	$\mathcal{A} + \mathcal{A}$	$I + 2M + S$

Table 3.1: Operations required for addition and doubling in $\mathbf{E}(\mathbb{F}_{2^m})$

Another option to improve the performance of multiplications is to first convert the scalar to another form so that the number of the necessary basic arithmetic operations is minimized. Examples of this are the Non-Adjacent-Form and the Mutual-Opposite-Form as discussed in detail in [8].
More recently, methods have been investigated to render ECC resistant against simple power analysis and other side channel attacks, as in [39], [12] or [21].

3.4.3 Implementations

There exist numerous works on the implementation of a cryptosystem based on elliptic curves. In [35] the EC equivalent of the ElGamal scheme and its implementation have been carefully studied. In [11] and [9] the implementation of elliptic curves over F_p and F_{2^m} in software are examined in great detail.
Nevertheless most of the work regarding implementations is focused on finding efficient hardware solutions. The smaller keysizes make ECC an interesting choice for smartcards or embedded systems as in [24] or [19].

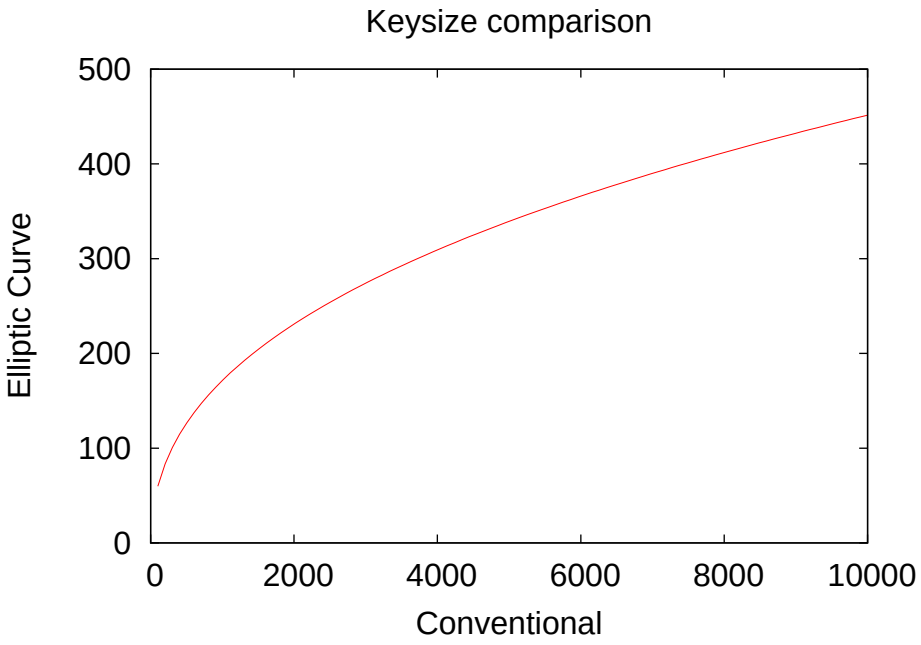

Figure 3.2: Key sizes for elliptic curve vs. conventional cryptosystems providing similar security

CHAPTER 4

The ECPP Protocol

In this chapter we construct the improved protocol. We give a review of the cryptographic primitives and subprotocols used to build the original PP protocol and explain all the necessary modifications we made to enhance it with elliptic curve cryptography.

4.1 Cryptographic Primitives

4.1.1 Oblivious Transfer

Oblivious Transfer protocols are cryptographic primitives that allow two parties to exchange information with the sender remaining oblivious on which part of the information was received. More formally in k-out-of-N-oblivious transfer (OT_N^k), the receiver can choose exactly k out of the N messages provided by the other party who does not learn which messages were chosen.
The concept was introduced by Rabin based on the RSA cryptoscheme and has then been further refined in order to find efficient protocols for secure multiparty computation. Because it can be shown that any polynomial time computable function can be securely evaluated using only oblivious transfers, it is considered one of the most critical problems in the field.

4.1.2 Yao Circuits

Yao- or garbled circuits, first introduced by Andy Yao in [47] can be used by two parties to securely evaluate every function that can be described as a boolean circuit, in a way, such that the inputs of the respective parties stay secret.
The basic idea is that one player creates encrypted circuits for all possible inputs of the other player who then can select the corresponding circuits using oblivious transfer and evaluate the respective results, which then may or may not be communicated to the

other player. In its original form, as used in the PP and $ECPP$ implementations, this scheme is secure against semi-honest adversaries.

4.1.3 Mixnet

Mixnets are cryptographic multiparty protocols used for anonymous communication. Messages are relayed through a series of proxies, that rearrange and randomize the input before transmitting it to the next player. A prominent example of such a network would be TOR [4].
In our case we use mixnets to deassociate events from their respective owners, before decrypting the matrix diagonal to get a time-sorted list of all events.

4.2 Review of the PP protocol

The PP protocol, as presented in [14], computes the BT predicate in a peer-to-peer fashion. It consists of several stages, implemented as subprotocols.

1. Distributed Key Generation
   All players jointly perform a distributed key generation protocol. They cooperatively generate a public key y and a shared secret key x.

2. Rank evaluations
   The players compute encryptions under y of the ranks (i.e. the position in a time-sorted list) of all events they hold. This is achieved by the players evaluating the GT predicate ($GT(e_i, e_j) := e_i(T) > e_j(T)$) for pairs of events, using a Yao circuit as described above.

3. Missing event evaluations
   All players jointly evaluate the ME predicate for all pairs of events to obtain encryptions of the number of missing events between them. They then fill a $N_{TE} \times N_{TE}$ matrix with the encrypted values gathered in steps 2 and 3.

4. Mixnet
   In order to deassociate the events from their respective owners, the players perform a mixnet over the input matrix. This means every player in turn performs a random permutation on the rows and columns of M. Exploiting the malleability of the cryptosystem, each player also randomizes all matrix elements by adding an encryption of zero to every element $m_{i,j}$, and then sends the permuted matrix to the next player.

5. Rank decryption
   The players now cooperatively decrypt the diagonal elements of the matrix, containing the rank information of the respective events. They can now identify time-consecutive events and sum up the encryptions of the corresponding number of missing events between them, obtaining an encryption of N_{ME} under y.

6. N_{ME} decryption
   The players now collaboratively decrypt N_{ME} and individually evaluate the presence of clones through equation 1.1.

4.3 Modifications

For the ECPP protocol we assume all players have previously agreed on using the same elliptic curve $\mathbf{E}$ of order $\#\mathbf{E}$ and a curve-point G, which should be a generator of the curve.

4.3.1 Distributed Key Generation

We do a straightforward adaptation of Pedersens distributed key generation protocol [42] to elliptic curve cryptography.
Each player i chooses a random value x_i and broadcasts a public key share x_iG to all other players. Additionally he computes random numbers $a_{i,1} \cdots a_{i,t-1}$ and sends each player a Shamir share of the private key, i.e. he evaluates the polynomial

$$f_i(y) = x_i + a_{i,1}y + \cdots + a_{i,t-1}y^{t-1} \tag{4.1}$$

of degree $t-1$ and sends the value $s_{i,j} = f_i(j)$ to each player j.
This results in all players sharing the same public key $H \in \mathbf{E}$, given by $\sum_i \chi_i G = \chi G$, with $\chi \in \{1, \ldots, (\#\mathbf{E} - 1)\}$ being the private key and every player i's share of it being $\chi_i = \sum_j s_{j,i}$.
To decrypt a message (A, B) each player i broadcasts $\chi_i A$. Through Lagrange interpolation each player can then recover χA, because we have:

$$\chi A = \sum_i x_i A = (\sum_i \sum_j s_{i,j} w_j)A = (\sum_j w_j \sum_i s_{i,j})A = \sum_j w_j \chi_j A \tag{4.2}$$

with w_j being the Lagrange weight $w_j = l_{0,j}$.

4.3.2 Cryptography

The subprotocols for encryption and decryption are standard ElGamal on elliptic curves (see 3.2) with one important caveat: To maintain the additive homomorphic property of the cryptosystem, the message embedding function has to be additively homomorphic as well. We use the function $f(m) := m \cdot G$ to encode a message, and rely on exhaustive searching over the message space for decryption. This is feasible because the message space in our case is sufficiently small, and decryption happens only in the very final stages of the protocol. The largest possible value that needs to be decrypted is the total number of missing events N_{ME}, which can not exceed $3 \cdot (N_{TE} - 1)$.

Encryption

To encrypt a message $m \in \{0, \ldots, \#\mathbf{E}-1\}$ a partner arbitrarily chooses a random integer $k \in \{1, \ldots, \#\mathbf{E}-1\}$ and computes the two curve points

$$A = kG, B = kH + mG \tag{4.3}$$

H denotes the public key generated with the above protocol. We call the pair (A, B) the cyphertext of the message m.

Decryption

After the recovery of the private key χ all partners can decrypt cyphertexts (A, B) to recover the encoded messages using the equation

$$B - \chi A = (mG + kH) - \chi kG = mG + \chi kG - \chi kG = mG \tag{4.4}$$

In order to decode a message, every partner enumerates all $nG, n \in \{0, \ldots, 3 \cdot (N_{TE}-1)\}$ until the correct message is found, i.e. $nG = mG$ [1].

Additive Homomorphism

For a cryptographic scheme to be additively homomorphic the encryption of the sum of two messages $E_k(m_1+m_2)$ must be equivalent to the sum of their respective cyphertexts $E_k(m_1) + E_k(m_2)$. For the presented scheme this property can be verified by observing

$$\begin{aligned} E_H(m_1) + E_H(m_2) &= (k_1G, k_1H + m_1G) + (k_2G, k_2H + m_2G) \\ &= (k_1G + k_2G, k_1H + k_2H + m_1G + m_2G) \\ &= ((k_1 + k_2)G, (k_1 + k_2)H + (m_1 + m_2)G) \quad = E_H(m_1 + m_2) \end{aligned}$$

with $k_1, k_2 \in \{0, \ldots, \#\mathbf{E} - 1)\}$. [2]

Malleability

Due to its probabilistic and homomorphic nature the presented scheme is also malleable, which means that a given cyphertext can be transformed into another encryption of the same plaintext m, without necessarily learning m.
This can be achieved by adding an encryption of 0 to the respective cyphertext. The resulting sum cannot be related to the original value in any way, due to the use of secret random variables in the encryption function.

[1] Alternatively each partner could precompute all possible values mG and use a lookup-table for decryption.
[2] The domainborders for k_1, k_2 have been corrected after the electronic submission of this thesis.

4.3.3 Oblivious Transfer

Oblivious transfers, used to evaluate the Yao circuits in stages 2 and 3 of the above protocol, consume the lion share of the protocols running time, and thus provide an appealing target for optimization. We present a protocol based on [41]. We assume both parties share two curve points $P_0 + P_1 = P$.

1. Bob selects a random $k \in \{1, \ldots, \#\mathbf{E} - 1\}$ and creates public keys (U_0, U_1) by computing $U_\sigma = kG$, $U_b = kG - P_\sigma$ and $U_{1-\sigma} = P_{1-\sigma} - U_b$. where σ is the index of the message Bob would like to receive.

2. Alice picks two random integers $r_0, r_0 \in \{1, \ldots, \#\mathbf{E} - 1\}$ and computes $V_0 = r_0 G$ and $V_1 = r_1 G$. She then computes $W_0 = r_0 U_0$ and $W_1 = r_1 U_1$ and sends to Bob $(V_0, V_1, X_0 = m_0 \oplus W_0, X_1 = m_1 \oplus W_1)$

3. Bob can now extract m_σ by computing $kV_\sigma = kr_\sigma G = W_i$ and then $W_i \oplus X_i = m_\sigma$.

Alice can verify the proper formulation of Bobs request by checking if $U_\sigma + U_{1-\sigma} = P$ and $U_\sigma - P_\sigma = -(U_{1-\sigma} - P_{1-\sigma})$. Bob cannot extract $m_{1-\sigma}$ because he cannot find n such that $U_{1-\sigma} = nG$. Alice cannot know which of (U_0, U_1) is equal to kG, therefore she stays oblivious on which message has been received by Bob.

4.4 The improved Protocol

Given the subprotocol modifications presented above, the improved PP protocol (ECPP) appears almost unchanged to the original protocol from [48]. Nevertheless please note the differences in the summation formulae in steps 2,3 and 4 respectively, implied by the additive nature of the group of elliptic curve points.

1. All Players jointly execute the above DKG subprotocol to generate a public key H and a shared private key χ for an instance of the additively homomorphic elliptic curve ElGamal cryptosystem.

2. For all pairs of events (e_i, e_j), $i \neq j$, the players owning events e_i and e_j jointly compute $E_H(GT(e_i, e_j)) := E_H([e_i(T) > e_j(T)]))$ and $E_H(GT(e_j, e_i))$ using a Yao circuit, with $E_H(1)$ or $E_H(0)$ mapped to the output wire if $e_i(T) > e_j(T)$ or $e_i(T) < e_j(T)$ respectively.
   After the circuit evaluation the player holding event e_i opens the output, computes $E_H(GT(e_j, e_i)) = E_H(1 - GT(e_i, e_j)) = E_H(1) - E_H(GT(e_i, e_j)$, and sends the result to the other player.

3. For all pairs of events (e_i, e_j), $i \neq j$, the players holding events e_i and e_j jointly evaluate $E_H(ME(e_i, e_j))$, the result of missing events for the pair (e_i, e_j), using a three-output Yao circuit. For the three outputs, we map $E_H(1)$ to 0, while we map $E_H(3)$, $E_H(2)$ and $E_H(1)$ to 1 for the first, second and third output respectively, corresponding to the number of missing events.
   The player holding event e_i opens the output of the circuit and computes an encryption under H of $ME(e_i, e_j)$ by summing up the individual output wires of the circuit as $E_H(\sum_{k=1}^{3} N_{ME,k}) = \sum_{k=1}^{3} E_H(N_{ME,k})$. (He does not share this result with the holder of event e_j.

4. For each of his events e_i, each player adds the bits he got in step 2, obtaining $E_H(RK(e_i))$, an encryption under H of the position of e_i in a time-sorted list of all considered N_{TE} events. Each player performs this operation locally, by calculating $E_H(RK(e_i)) = E_H(\sum_{j \neq i} GT(e_i, e_j)) = \sum_{j \neq i} E_H(GT(e_i, e_j))$. We can now imagine an N_{TE}-by-N_{TE} matrix formed by arranging the encrypted ME and filling the diagonal with the encrypted RK:

$$M = \begin{pmatrix} E_H(RK(e_1)) & \cdots & E_H(ME(e_i, e_{N_{TE}})) \\ \vdots & \ddots & \vdots \\ E_H(ME(e_{N_{TE}})) & \cdots & E_H(RK(e_{N_{TE}})) \end{pmatrix}$$

   Each line i of M corresponds to an event e_i, the element $M(i, i)$ to the rank of e_i, while element $M(i, j)$ denotes the number of missing events for the pair (e_i, e_j).

5. The players cooperatively perform a mixnet over M. To perform this, all players send the encryptions obtained in steps 2 and 4 to the player P_0, who then arranges them in the matrix M. Each player in turn, starting from P_0 randomly permutes

the matrix, and randomizes the matrix by adding $E_H(0)$ to each element $M(i,j)$, and sends the matrix to the next player. We denote the final result of the mixnet by M_n.

6. The players collaboratively decrypt the diagonal of M_n. It is now possible to identify two time-consecutive events, and therefore to identify and sum the corresponding ME results: If event e_i has rank u ($M_n(i,i) = u$) and event e_j has rank $u+1$ ($M_n(j,j) = u+1$), then $M_n(i,j)$ contains the number of missing events for the pair (e_i, e_j). By summing up the identified results, every player obtains an encryption under H of the total number of missing events N_{ME}.

7. The players collaboratively decrypt $E_H(N_{ME})$. Each partner P_i, based on his significance level α and misread probabilty p_{mr} can now evaluate the presence of clones through equation 1.1.

CHAPTER 5

Evaluation

In this chapter we give an analysis of the modified subprotocols in terms of security and complexity. Also, we give experimental performance evaluations of the ECPP protocol in various settings and compare them to the original implementation and implementations within the VIFF framework.

5.1 Security and Complexity

The round complexity of the new protocol remains unchanged compared to PP. It takes 3 rounds to evaluate the Yao circuits, and an additional n rounds to mix the matrix. The presented protocol is secure against computationally bound, passive adversaries.

5.2 Performance

In this section we give a performance comparison between the Java implementations of the original PP protocol, the ECPP protocol and implementations within the VIFF framework. We first present the methodology and the different testing scenarios, and then show the results.

5.2.1 Methodology

To measure performance we used the identical test setup as in [14]. It consists of up to ten identical machines with 3 GHz Pentium4 CPUs and 2GB of RAM each, which were connected through a 10/100 Mbs switch. The computers were configured with a 32-bit Fedora Core 7 Linux system (2.6.23). For the java protocols we used the 1.6.0 JRE from Sun, while the VIFF implementations were tested using version 1.0 of the VIFF framework [5].

To measure running times of the protocols we used the UNIX *time* command and bandwidth consumption was measured using the *tcpdump* [3] suite. To simulate network latency we used the *netem* kerneldriver [2], and limitations on CPU time were enforced using *cpulimit* [1].

5.2.2 Scenario

With one exception we run our tests in a setting that has been called the 'Real World Scenario', a model of a realistic supply chain. Every partner has a RCV- and a corresponding SHP-event, with the exception of the producer which has only one SHP event, and the retailer who only has a RCV event. Thus the total number of events in this setting is given by

$$N_{TE} = 2 \cdot n - 2 \tag{5.1}$$

with n denoting the number of partners.
Further, we analyze the impact of a larger number of events to the protocols, by running a campaign with 5 events per partner.

5.2.3 Security Parameters

To test our ECPP implementation we used the NIST recommended curves B163, B233, B283 and B409 [40]. Unless otherwise stated tests regarding the PP protocol were performed using a 1024-bit private key, whereas a 163-bit key was used for ECPP.
With the exception of figure 5.13 the VIFF protocol implementation was run using the VIFF passive runtime (providing security only versus passive adversaries).

5.2.4 Results

Keysize

In figure 5.1 we evaluate the impact of key length to the running time of the PP and the ECPP protocol for different numbers of players.

Performance Comparison

In figures 5.2 and 5.3 we compare timings and bandwidth consumption of PP, ECPP and the VIFF implementation for different numbers of partners.

Network Latency

A detailed analysis of the influence of network latency can be found in figures 5.4, 5.5, 5.6 where only one partner has a slow connection, and figures 5.7, 5.8, 5.9 where all partners experience network latency.

CPU Handicap

Figures 5.10 and 5.11 show running times for a scenario where one machine has a slow CPU, simulated by restricting the available user time of one partner to 50% and 10% respectively. Please note, that due to the multithreading support of the test CPUs the maximal value is 200%. The limitations therefore have a stronger impact to the effective running time of the ECPP protocol, because of its multithreaded architecture.

More events

Figure 5.12 shows timings for VIFF and ECPP with every partner holding 5 events.

Active Adversaries

We conclude the evaluation by showing time measurements of the VIFF implementation using a different runtime in which computations are secure against active adversaries. Figure 5.13.

5.2.5 Discussion

For equivalent security requirements we find the improved algorithm superior to the old implementation in terms of running time and asymptotic complexity. Because the relation between key lengths for elliptic curve and conventional cryptosystems providing similar security is exponential (equation 3.3, the relative performance increase due to ECC is also. A 409-bit key used with the new protocol provides security roughly comparable to that of a 8000-bit key in a conventional cryptosystems, but completes the computation in less than half of PP's time using only a 2048-bit key.
Since its running time grows linearly with the number of players, the presented protocol also outperforms the VIFF implementation for larger numbers of participants.
From the latency and CPU handicap simulations we can project that disturbing factors affect our protocol only linearly, whereas in Shamir sharing based approaches, idle-times grow exponentially with the total runtime of the protocol. It is further interesting to see how in a such an implementation a single player with limited resources has a larger impact on the overall performance compared to the presented peer-to-peer approach.
It it is possible to evaluate the Yao circuits for the GT and ME predicates in parallel, which is reflected in the multithreaded architecture of the provided implementation. The CPU handicap simulations therefore have a stronger effect on the ECPP simulations, but indicate how the possibility of parallelization could prove advantageous when run on a true multi-processor platform.

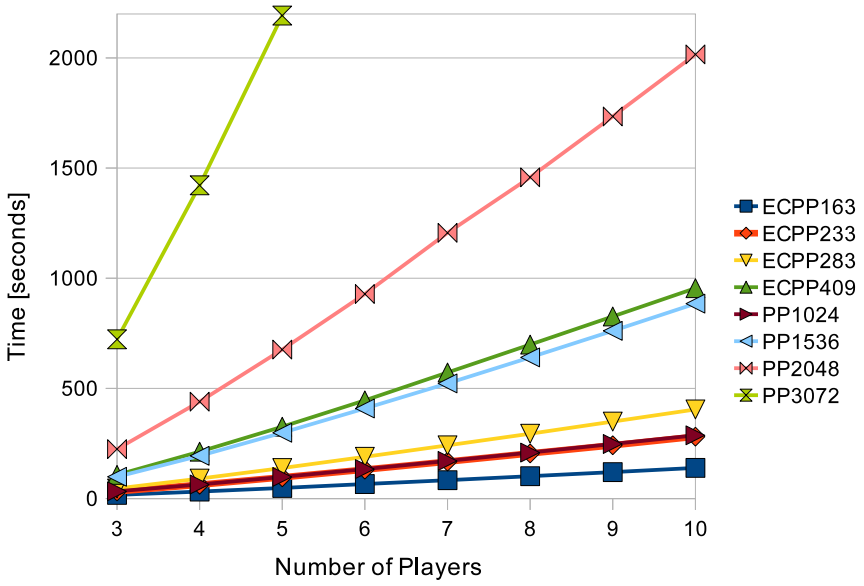

Figure 5.1: Running times of the PP and ECPP protocol with different keysizes

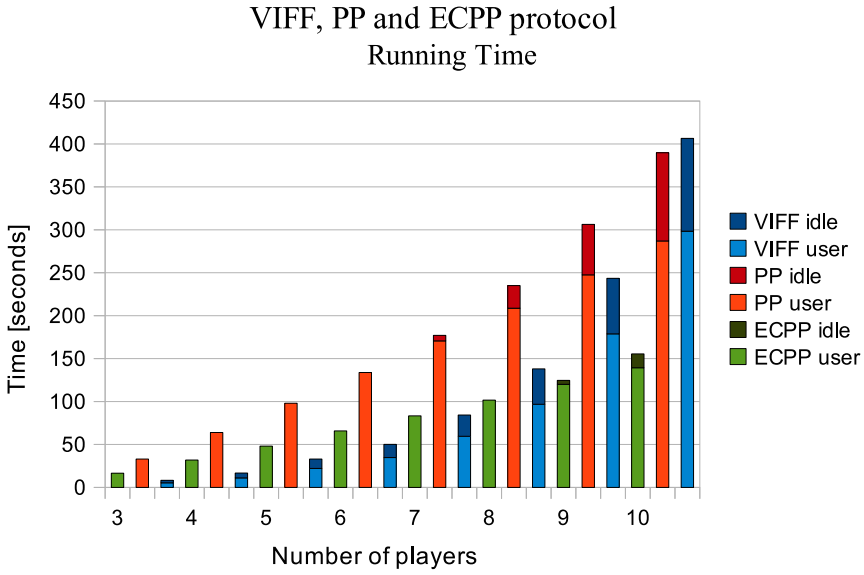

Figure 5.2: Detailed running times of the VIFF, PP and ECPP protocol

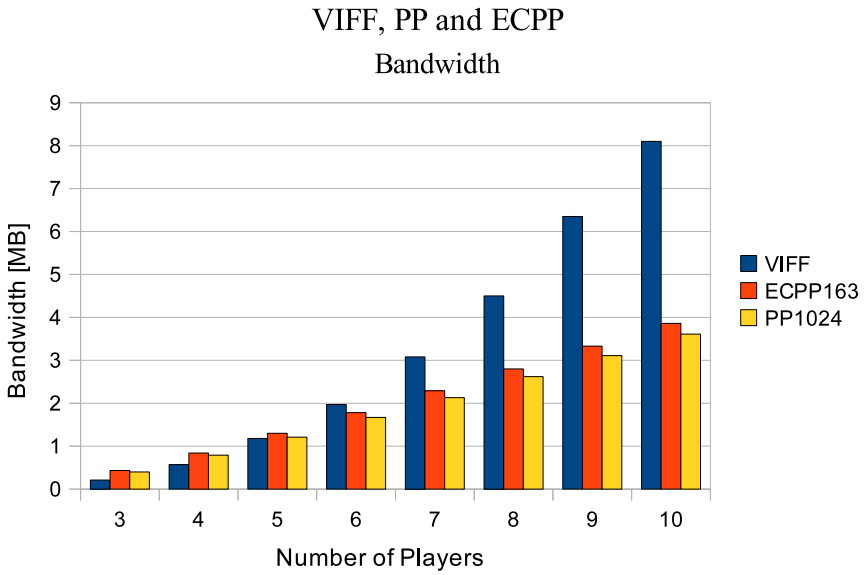

Figure 5.3: Bandwidth consumption of the VIFF, PP and ECPP protocol

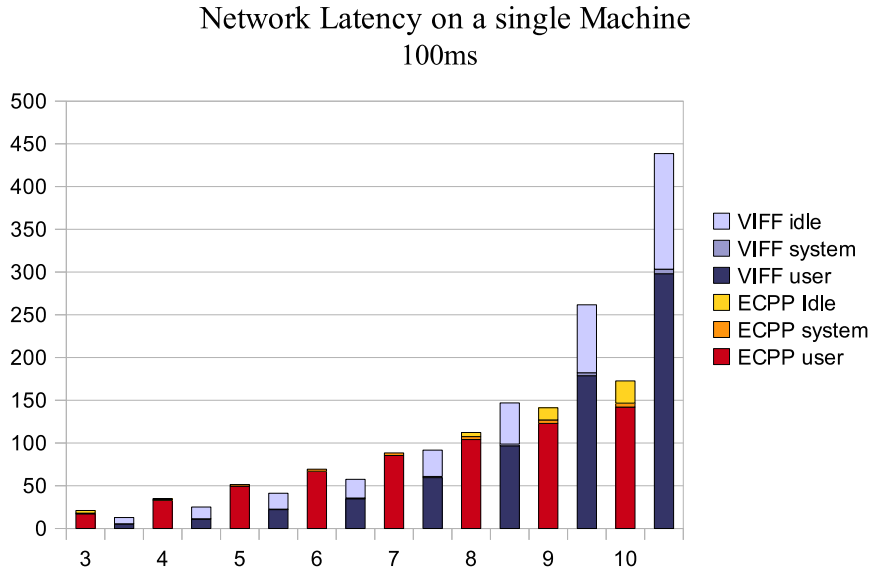

Figure 5.4: 100ms network latency on a single machine for VIFF and ECPP

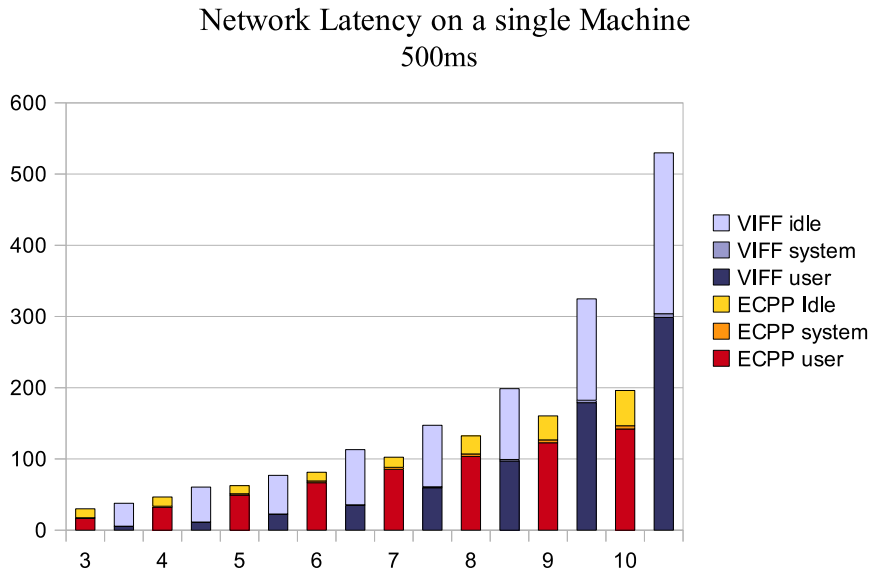

Figure 5.5: 500ms network latency on a single machine for VIFF and ECPP

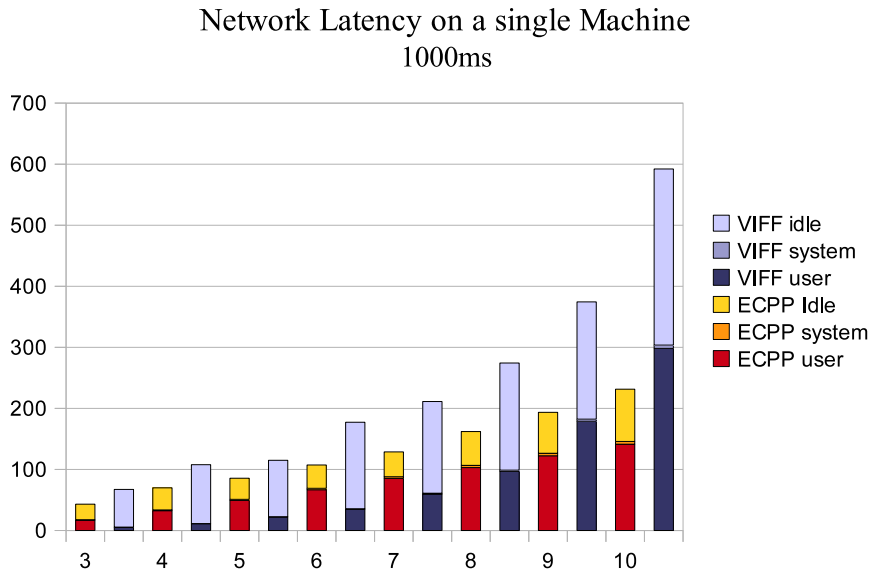

Figure 5.6: 1000ms network latency on a single machine for VIFF and ECPP

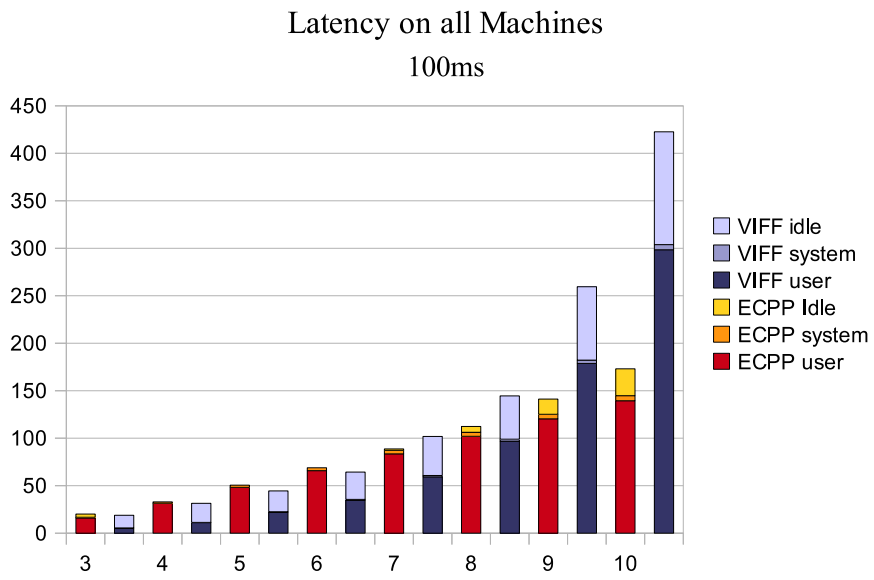

Figure 5.7: 100ms network latency on all machines for VIFF and ECPP

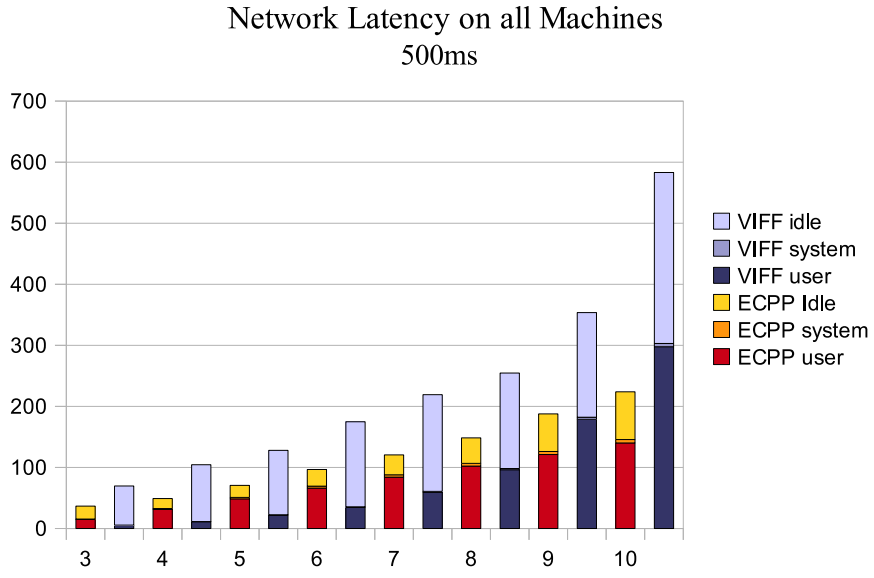

Figure 5.8: 500ms network latency on all machines for VIFF and ECPP

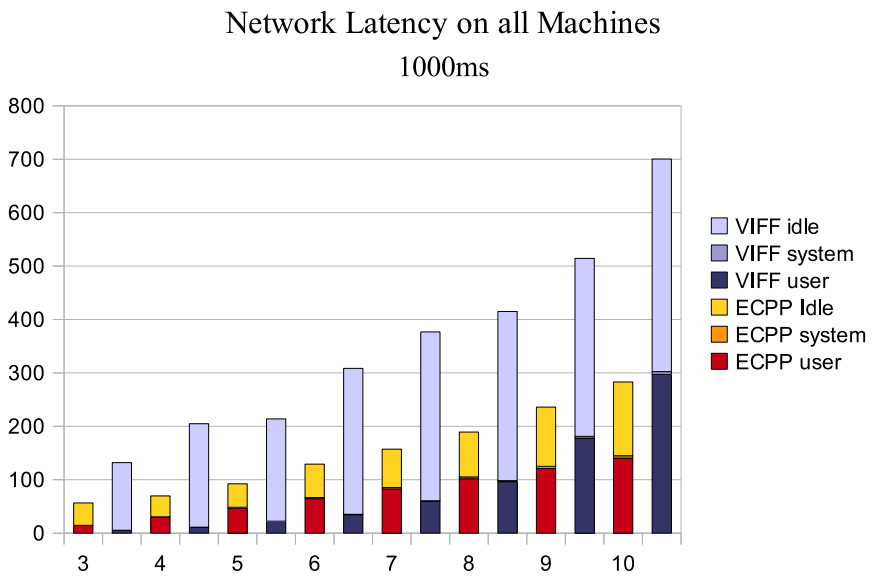

Figure 5.9: 1000ms network latency on all machines for VIFF and ECPP

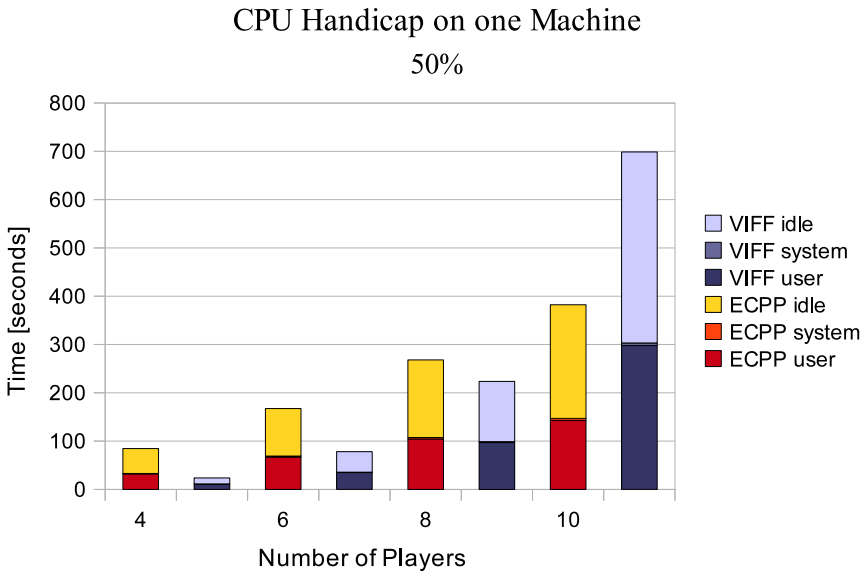

Figure 5.10: 50% CPU handicap on one machine for VIFF and ECPP

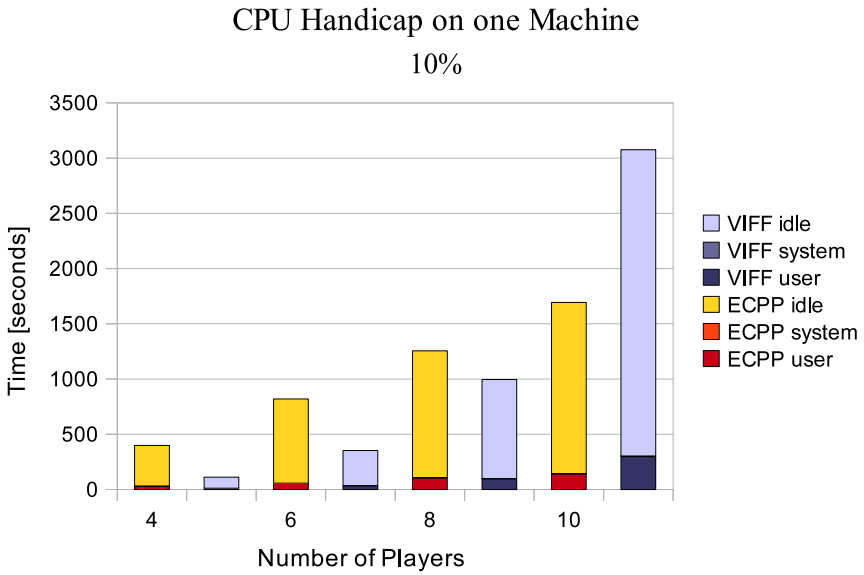

Figure 5.11: 10% CPU handicap on one machine for VIFF and ECPP

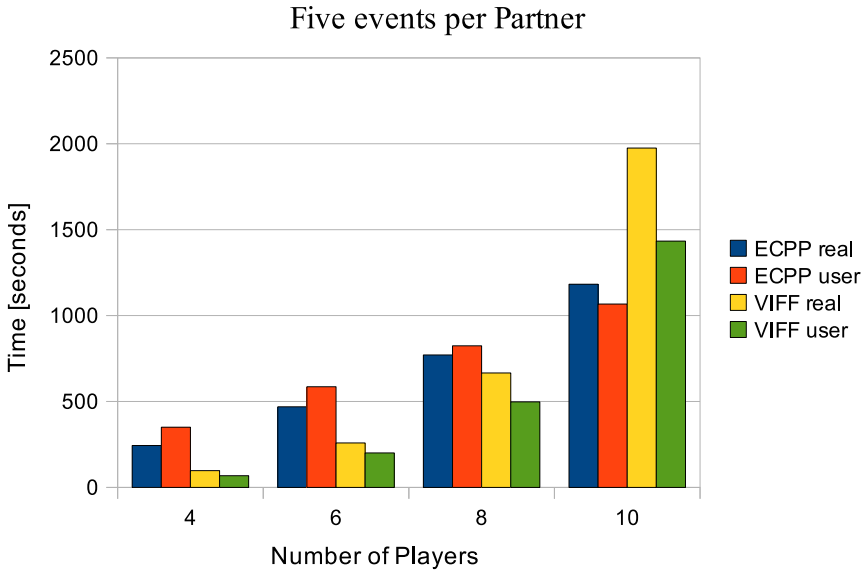

Figure 5.12: Running times for VIFF and ECPP with 5 events per player

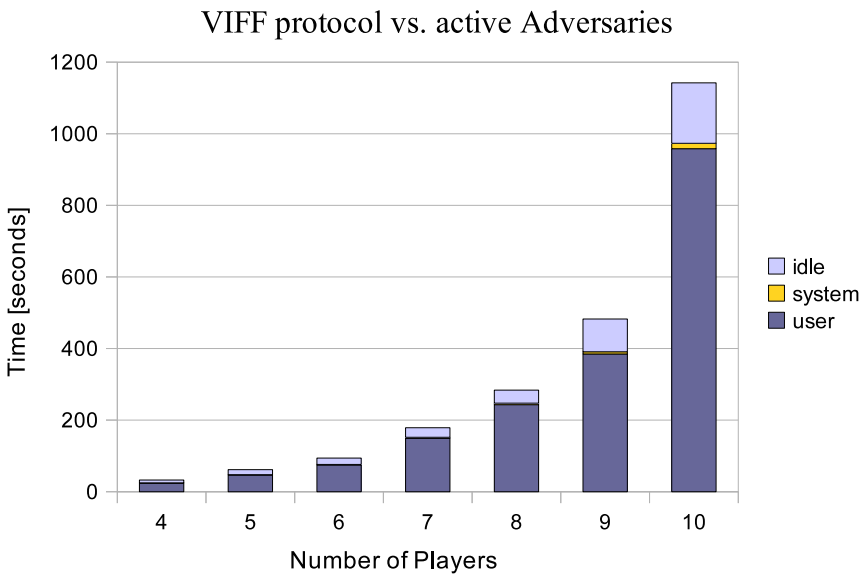

Figure 5.13: Timings of the VIFF implementation in a runtime secure against active adversaries

CHAPTER 6

Implementation

In this chapter we discuss the details of the implementation of the elliptic curve cryptosystem. We present the algorithms and representations we used for performing elliptic curve and finite field arithmetic.

6.1 Finite Field and Element Representation

For our implementation we focused on elliptic curves defined over finite fields of characteristic 2 ($\mathbf{E}(\mathbb{F}_{2^m})$). We represent field elements as binary polynomials $u(X) = \sum_i u_i X^i$ with coefficients $u_i \in \mathbb{F}_2$ modulo an irreducible polynomial of degree m. The polynomials themselves are represented by $(\lceil \frac{m}{t} \rceil)$ bit-vectors, where t denotes the wordsize of the underlying hardware. Such a representation can lead to very efficient arithmetic algorithms as we will show now.

6.2 Finite Field Arithmetic

Addition and Subtraction

For binary polynomials, addition and subtraction are equivalent. They can be implemented as a simple exclusive or (XOR) operation over the bit-vectors. Therefore we can save carry-bit handling and do not need an additional reduction step after the computation.

Squaring

To square a binary polynomial we can take advantage of the following observation:

$$u^2(X) = \sum u_i X^{2i} \tag{6.1}$$

This means that we can square a field element by simply inserting zeros into its internal bit-representation and reduce the result modulo the reduction polynomial. Additionally we apply a memory-time-tradeoff by precomputing a table of 2^{16} values containing the squares of all half-words to increase the speed of the zero-bit insertion process. We see that we can square a binary polynomial in time linear in the number of words of the internal representation, so the complexity of squaring is dominated by the complexity of the used reduction algorithm only.

Multiplication

The multiplication of two binary polynomials is the most frequent and therefore the most time critical operation used, when building an elliptic curve cryptosystem defined over $\mathbb{F}_{2^m}$. Several algorithms have been proposed and extensively studied. For our implementation we decided to apply a right-to-left comb method and converting the scalars to NAF.

Reduction

One of the most interesting features of the chosen field element representation is that very efficient by-word reduction algorithms can be applied if the hamming-weight of the reduction polynomial is small, and its 1-bits are relatively close to each other. See [9] for further detail.

Inversion

Given our representation the inversion/multiplication ratio in terms of time complexity is large on present day computers. This motivated the research to represent elliptic curve points in different coordinate systems, which lead to point-multiplication formulae avoiding inversions in the underlying field at the cost of additional multiplications and squarings.
In our implementation field inversions are only needed in the decryption stage. We therefore took a straight-forward approach to inversion, implementing an ordinary EGCD algorithm on binary polynomials.

6.3 Elliptic Curve Arithmetic

Coordinate System

As discussed in section 3.1 the choice of an appropriate coordinate system to represent curve points is of utmost importance to the performance of an elliptic curve cryptosystem. Our implementation is based on Lopez-Dahab coordinates, i.e. a curve-point is represented by the triple (x, y, z) representing the affine point $(x/z, y/z^2)$.

Point Addition and Doubling

We use the point addition formula from [23] taking $13M + 4S$, and the doubling formula from [32] taking $4M + 5S$. Please note that for all implemented curves $a_2 = 1$, saving one product.

Point Multiplication

Point multiplication is performed by first converting the scalar to Non-Adjacent-Form (NAF) to minimize the number of non-zero bits in its internal representation and then using a double-and-add algorithm to compute the multiple.

CHAPTER 7

Related Work

This work is primarily based on Davide Zanettis ideas summarized in [48]. In [14] protocols for privacy-preserving anti-counterfeiting were implemented in various SMC frameworks and a Java implementation of the PP protocol was given that served as a basis for the improved $ECPP$ protocol and in [46] Reto Weingart presented solutions to secure the PP protocol against active adversaries.
Another trace-based solution to the clone-detection problem can be found in [37], although the given solution is not suited to our problem. Partners are required to share data about the considered tags, therefore privacy preservation is not achieved. Another solution based on the concepts of machine learning can be found in [26]. Its main drawback being the need for a training stage, which makes this approach susceptible to changes in the topology of the supply chain.
Concludingly, in [45] an overview of tag-based methods can be found, i.e. methods that attack the problem by trying to detect and prevent the cloning of the RFID-tags themselves.
Other software implementations of elliptic curve cryptosystems can be found in [11], [9].

CHAPTER 8

Conclusion

In order to develop a privacy-preserving clone detection algorithm based on elliptic curve cryptography we performed the following work

8.1 Survey on elliptic curve cryptography

We gave an introduction to the concepts of elliptic curve cryptography and surveyed the existing research in the field. We listed numerous different representations of curve points and algorithms to perform arithmetics on elliptic curves and elements of the underlying finite field and compared them in terms of time-complexity.

8.2 Design and implementation of an improved protocol

We designed and implemented an additively homomorphic cryptographic scheme based on elliptic curves. We gave a list of modifications to the subprotocols of the existing solution to adapt it to the new scheme and presented a new protocol for privacy-preserving clone-detection in RFID-enabled supply chains which is secure against passive adversaries, based on elliptic curve cryptography.

8.3 Evaluation

We evaluated the performance of the new implementation in numerous scenarios and compared it to the previous protocol and an implementation in the VIFF SMC framework. We found our protocol superior to the former implementation in terms of running time, especially for larger keysizes. The peer-to-peer nature of the protocol proves to be advantageous over Shamir-sharing based solutions for larger numbers of players or events, and renders the protocol more insensitive to disruptive factors like network latency or partners with small CPU power.

Finally we gave an evaluation of a VIFF implementation which is secure against active adversaries.

8.4 Future Work

In [46] the PP protocol was extended to provide security even against active adversaries. The additional security requirements had significant impact on the complexity of the different subprotocols and thus furtherly increased the need for an efficient cryptographic scheme. A combination of this work with the methods developed here seems feasible and worthwhile.
Furthermore the development of a protocol that calculates the results of the BT predicate using a client-server approach was suggested in [14]. This could be beneficial in cases where the partners do not want to disclose even their involvement in such a computation to the other participants.

APPENDIX A

Code documentation

A.1 class ellipticcurve.EllipticCurve

Constructors

- EllipticCurve(**FiniteField2m** f, **BinaryPolynomial** a, **BinaryPolynomial** b)
  Create a new elliptic curve over the field f with curve parameters a and b

A.2 class ellipticcurve.EllipticCurvePoint

Constructors

- EllipticCurvePoint(**BinaryPolynomial** x, **BinaryPolynomial** y, **BinaryPolynomial** z)
  Create an elliptic curve point with projective coordinates (x, y, z)

- EllipticCurvePoint()
  Create an elliptic curve point with projective coordinates $(1, 1, 0)$

Arithmetic

- **void** doubling()
  Double this curve point

- **void** add(**EllipticCurvePoint** other)
  Add another curve point to this

- **void** sub(**EllipticCurvePoint** other)
  Subtract another curve point from this

- **EllipticCurvePoint** negative()
  Return the negative of this curve point
- **EllipticCurvePoint** diff(**EllipticCurvePoint** other)
  Return the difference between this and another curve point
- **EllipticCurvePoint** sum(**EllipticCurvePoint** other)
  Return the sum of another curve point and this
- **EllipticCurvePoint** multiple(**BigInteger** k)
  Return the multiple of this and a scalar k
- **BinaryPolynomial**[] toAffine()
  Return the affine coordinates of this curve point

Status

- **boolean** isPointAtInfinity()
  Returns true if this is the point at infinity

Private

- **byte**[] toNAF(**BigInteger** k)
  Return a byte array containing a representation of k in Non-Adjacent-Form (NAF)

A.3 class finitefield.FiniteField2m

Constructors

- FiniteField2m(**int** m, **ReductionAlgorithm** r, **BinaryPolynomial** p)
  Create a finite field of size 2^m with reduction polynomial p and a corresponding reduction algorithm r

Arithmetic

- **BinaryPolynomial** one()
  Return the binary polynomial 1
- **BinaryPolynomial** zero()
  Return the binary polynomial 0

A.4 class finitefield.BinaryPolynomial

Constructors

- BinaryPolynomial(**int** t, **BigInteger** value)
  Create a new binary polynomial of size $t \cdot 32$ bits and initialize it from value

Arithmetics

- **void** add(**BinaryPolynomial** other)
  Add another binary polynomial to this one
- **void** square()
  Square this binary polynomial and reduce it modulo the cryptosystems reduction polynomial [1]
- **void** mul(**BinaryPolynomial** other)
  Multiply this by another binary polynomial and reduce modulo the reduction polynomial
- **BinaryPolynomial** inverse(**BinaryPolynomial** other)
  Return the inverse of this modulo the reduction polynomial

Convenience functions

- **BinaryPolynomial** sum(**BinaryPolynomial** other)
- **BinaryPolynomial** squared()
- **BinaryPolynomial** product(**BinaryPolynomial** other)

Status

- **boolean** isZero()
  Return true if this is 0
- **int** degree()
  Return the degree of this polynomial

A.5 interface finitefield.ReductionAlgorithm

- **abstract void** reduce(**int**[] data)
- **BinaryPolynomial** reductionPolynomial

[1] The reduction polynomial was made a global parameter in order to save bandwidth consumption

APPENDIX B

NIST curves

To evaluate our implementation we used the following four curves recommended by the national institute for standards and technology [40]. For all these curves $a = 1$. We also give the reduction polynomial $R(x)$ of the underlying field and a generator G in affine coordinates. Field elements are represented using a polynomial basis.

B-163

$$
\begin{aligned}
R(x) &= x^{163} + x^7 + x^6 + x^3 + 1 \\
b &= \texttt{2 0a601907} \\
&\quad \texttt{b8c953ca 1481eb10 512f7874 4a3205fd} \\
G_x &= \texttt{3 f0eba162} \\
&\quad \texttt{86a2d57e a0991168 d4994637 e8343e36} \\
G_y &= \texttt{0 d51fbc6c} \\
&\quad \texttt{71a0094f a2cdd545 b11c5c0c 797324f1}
\end{aligned}
$$

B-233

$$
\begin{aligned}
R(x) &= x^{233} + x^{74} + 1 \\
b &= \texttt{066 647ede6c 332c7f8c 0923bb58} \\
&\quad \texttt{213b333b 20e9ce42 81fe115f 7d8f90ad} \\
G_x &= \texttt{0fa c9dfcbac 8313bb21 39f1bb75} \\
&\quad \texttt{5fef65bc 391f8b36 f8f8eb73 71fd558b} \\
G_y &= \texttt{100 6a08a419 03350678 e58528be} \\
&\quad \texttt{bf8a0bef f867a7ca 36716f7e 01f81052}
\end{aligned}
$$

B-283

$$
\begin{aligned}
R(x) &= x^{283} + x^{12} + x^7 + x^5 + 1 \\
b &= \texttt{27b680a} \\
&\quad \texttt{c8b8596d a5a4af8a 19a0303f ca97fd76} \\
&\quad \texttt{45309fa2 a581485a f6263e31 3b79a2f5} \\
G_x &= \texttt{5f93925} \\
&\quad \texttt{8db7dd90 e1934f8c 70b0dfec 2eed25b8} \\
&\quad \texttt{557eac9c 80e2e198 f8cdbecd 86b12053} \\
G_y &= \texttt{3676854} \\
&\quad \texttt{fe24141c b98fe6d4 b20d02b4 516ff702} \\
&\quad \texttt{350eddb0 826779c8 13f0df45 be8112f4}
\end{aligned}
$$

B-409

$$
\begin{aligned}
R(x) &= x^{409} + x^{87} + 1 \\
b &= \texttt{021a5c2} \\
&\quad \texttt{c8ee9feb 5c4b9a75 3b7b476b 7fd6422e} \\
&\quad \texttt{f1f3dd67 4761fa99 d6ac27c8 a9a197b2} \\
&\quad \texttt{72822f6c d57a55aa 4f50ae31 7b13545f} \\
G_x &= \texttt{15d4860} \\
&\quad \texttt{d088ddb3 496b0c60 64756260 441cde4a} \\
&\quad \texttt{f1771d4d b01ffe5b 34e59703 dc255a86} \\
&\quad \texttt{8a118051 5603aeab 60794e54 bb7996a7} \\
G_y &= \texttt{061b1cf} \\
&\quad \texttt{ab6be5f3 2bbfa783 24ed106a 7636b9c5} \\
&\quad \texttt{a7bd198d 0158aa4f 5488d08f 38514f1f} \\
&\quad \texttt{df4b4f40 d2181b36 81c364ba 0273c706}
\end{aligned}
$$

BIBLIOGRAPHY

[1] `http://cpulimit.sourceforge.net/`.

[2] `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`.

[3] `http://www.tcpdump.org/`.

[4] `http://www.torproject.org`.

[5] `http://www.viff.dk/`.

[6] S. Vanstone A. Menezes, P. van Oorschot. *Handbook of applied cryptography*. CRC Press, 1996.

[7] Al-Daoud, Mahmod, Rushdan, and Kilicman. A new addition formula for elliptic curves over GF(2^n). *IEEETC: IEEE Transactions on Computers*, 51, 2002.

[8] P. Balasubramaniam and E. Karthikeyan. Elliptic curve scalar multiplication algorithm using complementary recoding. *Applied Mathematics and Computation*, 190(1):51–56, July 2007.

[9] M. Brown, D. Hankerson, J. Lopez, and A. Menezes. Software implementation of the nist elliptic curves over prime fields, 2000.

[10] D. V. Chudnovsky and G. V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Technical Report RC 11261, Mathematics Dept., Columbia University, 1985.

[11] Alfred Menezes Darrel Hankerson, Julio Lopez Hernandez. Software implementation of elliptic curve cryptography over binary fields. In *Lecture notes in computer science*, pages 1–24. Springer-Verlag, 2000.

[12] Dimitrov, Imbert, and Mishra. Efficient and secure elliptic curve point multiplication using double-base chains. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 2005.

[13] Edwards. A normal form for elliptic curves. *BAMS: Bulletin of the American Mathematical Society*, 44, 2007.

[14] L. Fellmann. Privacy-preserving anti-counterfeiting for rfid-enhanced supply chains, 2009.

[15] Fong, Hankerson, Lopez, and Menezes. Field inversion and point halving revisited. *IEEETC: IEEE Transactions on Computers*, 53, 2004.

[16] Organization for economic Co-operation and Development. The economic impact of counterfeiting and privacy. 2005.

[17] Gallant, Lambert, and Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *CRYPTO: Proceedings of Crypto*, 2001.

[18] Guajardo and Paar. Efficient algorithms for elliptic curve cryptosystems. In *CRYPTO: Proceedings of Crypto*, 1997.

[19] Nils Gura, Arun Patel, Arvinderpal W, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. pages 119–132, 2004.

[20] T. Ono H. Cohen, A. Miyaji. Efficient elliptic curve exponentiation using mixed coordinates. *Advances in Cryptology, ASIACRYPT 98*, pages 51–65, 1998.

[21] Mustapha Hedabou, Pierre Pinel, and Lucien Beneteau. A comb method to render ecc resistant against side channel attacks, 2004.

[22] B. Hess. Clone detection in rfid-enhanced supply chains, 2009.

[23] Higuchi and Takagi. A fast addition algorithm for elliptic curve arithmetic in GF(2^n) using projective coordinates. *IPL: Information Processing Letters*, 76, 2000.

[24] Yvonne Hitchcock, Edward Dawson, Andrew Clark, and Paul Montague. Implementing an efficient elliptic curve cryptosystem over gf (p) on a smart card, 2003.

[25] N. Smart I. Blake, G. Seroussi. *Elliptic curves in cryptography*. Cambridge University Press, 1999.

[26] Brian King and Xiaolan Zhang 0003. Securing the pharmaceutical supply chain using RFID. pages 23–28. IEEE Computer Society, 2007.

[27] Knudsen. Elliptic scalar multiplication using point halving. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1999.

[28] Koblitz, Menezes, and Vanstone. The state of elliptic curve cryptography. *IJDCC: Designs, Codes and Cryptography*, 19, 2000.

[29] N. Koblitz. Ellptic curve cryptosystems. *Mathematics of computation*, 48:203–209, 1987.

[30] Koyama, Maurer, Okamoto, and Vanstone. New public-key schemes based on elliptic curves over the ring Z_n. In *CRYPTO: Proceedings of Crypto*, 1991.

[31] Koyama and Tsuruoka. Speeding up elliptic cryptosystems by using a signed binary window method. In *CRYPTO: Proceedings of Crypto*, 1992.

[32] Lopez and Dahab. Fast multiplication on elliptic curves over GF(2^m) without precomputation. In *CHES: International Workshop on Cryptographic Hardware and Embedded Systems, CHES, LNCS*, 1999.

[33] J. Lopez and R. Dahab. Improved algorithms for elliptic curve arithmetic in GF(2^n). *Lecture Notes in Computer Science*, 1556:201–212, 1999.

[34] Menezes, Okamoto, and Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEETIT: IEEE Transactions on Information Theory*, 39, 1993.

[35] A. J. Menezes and S. A. Vanstone. Elliptic curve cryptosystems and their implementation. *Journal of Cryptology*, 6:209–224, 1993.

[36] V. Miller. Uses of elliptic curves in cryptography. In *Advances in Cryptology, CRYPTO 85*, pages 417–426. Springer-Verlag, 1986.

[37] L. T. Mirowski and J Hartnett. Deckard: a system to detect change of RFID tag ownership. pages 89–98, 2007.

[38] Miyaji, Ono, and Cohen. Efficient elliptic curve exponentiation. In *ICIS: International Conference on Information and Communications Security (ICIS), LNCS*, 1997.

[39] Moller. Securing elliptic curve point multiplication against side-channel attacks. In *ISW: International Workshop on Information Security, LNCS*, 2001.

[40] National Institute of Standards and Technology. Recommended elliptic curves for federal goverment use. 1999.

[41] Parakh. Oblivious transfer using elliptic curves. *CRYPTOLOGIA: Cryptologia*, 31, 2007.

[42] T.P. Pedersen. A threshold cryptosystem without a trusted party. In *Lecture notes in computer science, EUROCRYPT*, pages 522–526. Springer-Verlag, 1991.

[43] J. M. Pollard. Monte Carlo methods for index computation (mod$p$). *Mathematics of Computation*, 32(143):918–924, July 1978.

[44] Schroeppel, Orman, O'Malley, and Spatscheck. Fast key exchange with elliptic curve systems. In *CRYPTO: Proceedings of Crypto*, 1995.

[45] Thorsten Staake, Frédéric Thiesse, and Elgar Fleisch. Extending the epc network: the potential of rfid in anti-counterfeiting. In *SAC*, pages 1607–1612, 2005.

[46] R. Weingart. A privacy-preserving clone detection mechanism against realistic adversaries, 2010.

[47] Yao. How to generate and exchange secrets (extended abstract). In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 1986.

[48] Davide Zanetti, Leo Fellmann, and Srdjan Čapkun. Privacy-preserving clone detection for RFID-enabled supply chains. In *IEEE RFID '10: Proceedings of the 2010 IEEE International Conference on RFID*, 2010.