ENCODING EFFICIENT ATTRIBUTES USING PRIME MODULO METHOD FOR ANONYMOUS CREDENTIALS

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Sayantica Pattanayak

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Electrical and Computer Engineering

January 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Encoding Efficient Attributes Using Prime Modulo Method for Anonymous Credentials

By

Sayantica Pattanayak

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr Debasis Dawn
Chair

Dr Raj Katti

Dr Kenneth Magel

Approved:

January 29, 2015
Date

Dr Scott Smith
Department Chair

ABSTRACT

The thesis explains a new method of encoding binary and finite set attributes in the anonymous credential system. To encode each binary and finite set attribute, we used the Chinese remainder theorem instead of using prime numbers [1].We then used the divisibility and coprime properties to efficiently prove the presence and absence of the attributes. The system is built on strong RSA assumptions. The new method can incorporate large numbers of binary and finite set attributes as compared to the Camenisch-Groβ credential system.Our new method can be used in electronic cards, health insurance cards and also in small devices like smart cards and cell phones.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Raj Katti for his continuous support, patience, motivation and immense knowledge. I could not have imagined having a better advisor and mentor for my Master Program. Besides my advisor I would like to thank Dr. Debasis Dawn for being my coadvisor.I would also like to thank Dr. Kenneth Magel to be in my Supervisory committee. Thanks to Seyed for helping me in writing my thesis.

I would also like to thank my husband and my three-year-old son. They were always there cheering me up and encouraging me with their best wishes. Thanks to the faculty and staff of the Electrical and Computer Engineering department of NDSU for providing such an excellent environment to learn and progress.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

As new technology is radically advancing our freedom, it is also helping our adversaries to invade our data privacy. Privacy concerns exist wherever our personal and sensitive information is collected and stored in digital form. Improper or nonexistent disclosure control can be the root cause of privacy issues. Data privacy issues can be found in a wide range of sources, such as electronic identity cards, medical records, transactions in financial institutions and privacy breaches. There is a need to defend our own information from unauthorized access, use, disclosure, disruption, modification, perusal, recording, inspection and destruction.

The challenge in data privacy is to share data while protecting personally identifiable information. The field of data security design utilizes software, hardware and human resources to address these issues. As the laws and regulations related to data protection are constantly changing, it is important to keep abreast of any changes in the law and continually reassess our compliance with data privacy and security. In order not to give away too much personal information, e-mails should be encrypted and browsing of webpages as well as other online activities should be done traceless. Nowadays a major issue with privacy relates back to social networking. For example, there are millions of users on Facebook. People may be tagged in photos or have valuable information exposed about themselves either by choice or, most of the time, unexpectedly by others. It is important to be cautious of what is being said over the Internet and what information is being displayed, as well as photos, because this all can be searched across the Web and used to access private databases, making it easy for anyone quickly to go online and look at a person's profile. To raise awareness among people about data privacy, January 28 is celebrated as Data Privacy Day.

Since the early days of communication, diplomats and military commanders have understood that it is necessary to provide some mechanism to protect the confidentiality of correspondence and to have some means of detecting tampering. Julius Caesar, who is credited with the invention of the Caesar Cipher c. 50 B.C., created the cipher in order to prevent his secret messages from being read if they fell into the wrong hands, but for the most part protection was achieved through the application of procedural handling controls. Sensitive information was marked up to indicate that it should be protected and transported by trusted persons, guarded and stored in a secure environment or strong box. As postal services expanded, governments created official organizations to intercept, decipher, read and reseal letters.

Electronic Identity cards (eID) have been widely used to authorize access to buildings, web services, use of facilities, etc. The eID cards are government issued identity cards for offline and online identification and are used by trusted organizations (government, company or university) for services provided by the organization. Belgium is the first European country to start using the eID cards. The eID cards contain several attributes of the user which are highly privacy-sensitive. By attributes we mean nationality, sex, civil status, hair and eye color, and applicable minority status, such as blind, partially sighted, wearing corrective lenses or hearing impaired. One of the serious issues in eID cards is the user's privacy, and many people are unaware of their pitfalls. Careless use of eID cards can cause economic and psychosocial damage. Each person is given a separate national registry number (NRN) for their eID cards. Once the NRN is known, then the user's different attributes can be revealed. Hence, there is a need for strong security and privacy protection.

Similarly as medical records are converted into an electronic format, the risk of compromising a patient's data is increasing. A person may not wish his or her medical records to

be revealed to others. This may be because they are concerned that it might affect their employment, or it may be because they would not wish for others to know about medical or psychological conditions or treatments which would be embarrassing; a pharmacist doesn't need to know the medical history of a patient to give medicine. Revealing medical data could also reveal other details about one's personal life. There are three major categories of medical privacy: informational (the degree of control over personal information), physical (the degree of physical inaccessibility to others), and psychological (the extent to which the doctor respects patients' cultural beliefs, inner thoughts, values, feelings, and religious practices and allows them to make personal decisions). Physicians and psychiatrists in many cultures and countries have standards for doctor-patient relationships which include maintaining confidentiality. In some cases, the physician-patient privilege is legally protected. These practices are in place to protect the dignity of patients and to ensure that patients will feel free to reveal complete and accurate information required for them to receive the correct treatment. The United States has laws governing privacy of private health information: HIPAA and the HITECH. The authors in [24] describe how the credential system is important in the health care industry. However if the credential is used more than once, the anonymity of the user is revoked.

At the same time the move to the electronic or digital approach in eID, medical records, etc. gave us an opportunity to improve data privacy by enhancing cryptographic techniques. Using a credential system, an organization can verify a person's information without revealing any secret data or attributes to the organization. In the paper-based world, a credential is like a passport, driver's license, or voter's identity card. However, in the digital world a credential is in the form of digital signature. A credential system is anonymous if the user is allowed to show the credential to the verifier without revealing any other information to the verifier. A verifier can

infer nothing about which the user is other than that the user has the right credential. This system allows users to selectively prove statements about their identity attributes. In [2] Camenisch and Lysyankaya proposed a signature scheme based on RSA (Rivest Shamir Adleman) assumptions. The authors have shown how to issue a signature on the committed value and how to prove the knowledge of the signature on the committed value. The authors in [20] proposed a method so that a credential cannot be copied and shared. The Scheme is based on DAA (Direct Anonymous Attestation)[21]. DAA is an anonymous digital signature scheme which provides the signers privacy and authentication. Canard and Lescuyer [22] uses sanitizable signatures to alter some parts of the signed message in such a way that the message can still be verified without the actual signer. The authors in [22] not only proposed an anonymous credential system but also a way to revoke the credential if someone misuses the credential.

Camenisch and Lysyankaya [3] proposed a credential system which is anonymous and based on the signature scheme these authors proposed in [2]. With anonymous credentials the user has to randomize the credential before sending it to the verifier in order to prevent revealing data in the credential [19]. The authors in [3] were the first to propose a solution where the users are allowed to demonstrate the credential as many times as they want without showing the actual credential. However, the complexity of credentials is linear by the total number of attributes.

Camenisch and Groβ [1] used the signature scheme proposed by Camenisch and Lysyankaya[2] and extended the credential system introduced by Camenisch and Lysyankaya[3] to incorporate finite set attributes in one base. The idea behind the work is to encode all the binary and finite set attributes as prime numbers and prove the presence and absence of the attributes without showing the attributes to the verifier. The computational complexity of the proofs does not depend on the number of attributes present in the credential.

Here, we propose a new solution to incorporate the finite and binary set attributes in one base using the Chinese remainder theorem. For each attribute set *i* we choose a prime moduli e_i. All the attributes are assigned to E where Emod e_i= the encoding of the attribute value and $E<\Pi_{i=1}^{t} e_i$ (t is the total number of attribute sets).Our proofs provide a highly efficient toolkit of attribute proofs, as well as AND, NOT and OR proofs over binary or finite set attributes. Our proof shows that the number of bits required to encode the binary or finite set attributes is much less than the one proposed by Camenisch and Lysyankaya [1].

The thesis contains five chapters. Chapter two includes the definitions and assumptions which are required in our work. Chapter three explains the Camenisch-Lysyankaya signature and proving the knowledge of the signature. Chapter four describes the prime encoding method by Groβ-Camenisch [1]. Chapter five includes our method of encoding the credential i.e. the Prime Modulo Encoding Method and detailed proof of the credential. Chapter six includes comparisons of our method with the prime encoding method.

CHAPTER 2. PRELIMINARIES

2.1. Assumptions

Special RSA Modulus: We call a modulus n a special RSA modulus if it has the form pq, where p=2q'+1 and q=2q'+1 are safe primes. We call this setup special RSA (SRSA) setting.

Hardness of Factoring: This is a Strong RSA based Assumption. This assumption states that the RSA problem is intractable even when the solver is allowed to choose the public exponent e (for e>1). More precisely, given a modulus n of unknown factorization, and g$\in$ Z_n^*, it is hard to compute h$\in$ Z_n^* and e, such that $h^e \equiv g \bmod n$.The strong RSA assumption implies that the factoring is hard. The strong RSA assumption was first used for constructing signature schemes provably secure against existential forgery without resorting to a random oracle.

Discrete Logarithm (DL) Assumption [4]: This is a computational difficulty assumption based on cyclic groups. Let G be a cyclic group of order n with g $\in$ G be a generator of discrete logarithm (DL) in G is

Given a $\in$ G, compute x ($0 \leq x < |G|$) such that $h=g^x$.

x is called the discrete logarithm of a with respect to base; here we assume that the discrete logarithm problem is hard; given the above set up it is hard to find the integer x.

2.2. Integer Commitments

The receiver chooses two large primes; p and q .The receiver also chooses a generator g from group G of prime order q. Here p,q,g,g_0 are public parameters. To commit to the value v $\in$ Z_q, sender picks a random r $\in$ Z_q and set $C=Com(v,r)=g_0^r * g^v$. The commitment schemes are best described by the authors in [4] [5].

2.3. Sigma Protocol [12]

Common Input: P (prover) and V (verifier) both have x

Private input: P has w such that (x, w) Є relations

V sends a random string e

P sends a reply Z

V accepts based solely on (x,a,e,z)

2.4. Zero Knowledge Proofs based on Discrete Logarithm (DL)

In the following we assume that group G=<g> of large known order Q and second generator h whose DL to the base g is not known. We define the DL y to the base g as $y=g^x$. The following is a brief review of different types of proof of knowledge that we are going to use.

Proving the knowledge of DL x of group element y to the base g [7] [8]. The prover chooses a random r from $Z_{q\ and}$ compute $t=g^r$ and sends it to the verifier. The verifier picks a random challenge c and sends it to the prover. The prover computes s=r-cx and sends it to the verifier. The verifier accepts if and only if $g^s y^c=t$.

Proving the knowledge of representation of an element y to the bases $\{g_1, g2_{\ldots}g_l\}$ [9] [10], i.e. proving the knowledge of representation of $\{x_1, x2_{\ldots}x_l\}$. The prover chooses a random $\{r_1, r2_{\ldots}r_l\}$ from Z_q and computes $t=\prod_{j=1}^{r} gj^{\,rj}$ and sends it to the verifier. The verifier picks a random challenge c and sends it to the prover. The prover computes s_j=rj-cxj and sends it to the verifier. The verifier accepts if $g^s y^c=t$.

Proving the equality of the discrete logarithms of the elements y_1 and y_2 to the bases g and h, respectively [13]. Let $y_1=g^x$ and $y_2=h^x$. The prover chooses a random r Є Z_q^*, computes $t_1=g^r$, $t2=h^r$, and sends t_1 and t_2 to the verifier. The verifier picks a random challenge c and sends it

to the prover. The prover computes s =r-cx and sends s to the verifier. The verifier accepts if $g^{s1}y_1^c = t_1$ and $g^{s2}y_1^c = t_2$ holds. This protocol is denoted by PK{(α) : $y_1 = g^{\alpha} \Lambda y_2 = y_1^{\alpha}$}

Proving that the discrete logarithm lies in a given range [14]. Here we have to prove that the discrete logarithm x of y to the base g satisfies $2^{l1}-2^{l2}<x<2^{l1}+2^{l2}$ for a given parameter l_1 and l_2.The parameter 2^{l1} acts as an offset and can be taken as Zero. The authors in [4] [18] describe the protocol in detail. A brief overview of the protocol is given below. The protocol is denoted as

PK {(α): $y=g^{\alpha} \Lambda 2^{l1}- 2^{\epsilon l2+2}<\alpha<2^{l1}+2^{\epsilon l2+2}$}

Prover		Verifier
(g,Q,y,x)		(g,Q,y)
$r \epsilon_r \{-2^{\epsilon l2}……..2^{\epsilon l2}\}$		
$t:=g^r$ sends	t →	t
		←c{0,1}
$s:=r-c(x-2^{l1})$ sends s→		s
		$-2^{\epsilon l2+1} < s<-2^{\epsilon l2+1}$
		$t = g\hat{}(s - c2^{l1})\ y^c$

When repeated a number of times, the prover can convince the verifier that the secret x lies with the given range $2^{l1}- 2^{\epsilon l2+2}<x<2^{l1}+ 2^{\epsilon l2+2}<$ for given parameters $\epsilon\ l_1,l_2$.

CHAPTER 3. CAMENISCH-LYSYANKAYA SIGNATURE

3.1. Description of Camenisch-Lysyankaya Signature (CL Signature)

Let l_m, l_e, l_n, l_r, and L be the system parameter and L_r be the security parameter.

Message space: This is the set $\{(m_0 \text{ and } m_1): m_i \epsilon \pm \{0, 1\}^{lm}\}$

Signing Algorithm: On input of m_0 and m_1 choose a random prime number e of length $l_e > l_m + 2$ and a random number v of length $l_v = l_n + l_m + l_r$. Compute

$$A = (z/R_0^{m0} R^{m1}{}_1 S^v)^{1/e} \bmod n$$

The signature consists of (*e*,A,v)

Verification Algorithm: To verify that the tuple (*e*,A,v) is a signature on message (m_0 and m_1), check that the following statement holds.

$Z \equiv \pm R_0^{m0} R_1^{m1} A^e S^v \pmod n$, $m_i \epsilon \pm\{0,1\}^{lm}$ and $2^{le} > e > 2^{le-1}$

3.2. Proving the Knowledge of the Signature

In CL signature, if we make A public that would destroy privacy, as that would make all transactions linkable. Hence, A was randomized. Given a signature (A,*e*,v), the tuple ($A' = AS^{-r} \bmod n$, *e*,$v' = v + er$) is also a valid signature, provided that $A \in \langle S \rangle$ and r is chosen uniformly at random from $\{0,1\}^{ln+l\Phi}$. Thus, the user could compute a fresh A' each time, reveal it and then run the following protocol with the verifier.

PK $\{(\varepsilon, v', \mu_0, \ldots \mu_L)$:

$Z \equiv \pm R_0^{\mu 0} R_1^{\mu 1} A'^{\varepsilon} S^{v'} \pmod n \; \Lambda$

$\Lambda \; \mu_i \epsilon \pm\{0,1\}^{lm} \; \Lambda \; \varepsilon \epsilon [2^{le-1} + 1, 2^{le} - 1]\}$

The secret μ_i and ϵ should actually lie in the small interval, i.e. the signer needs to choose *e* from $[2^{le-1} - 2^{le'} + 1, 2^{le-1} + 2^{le'} - 1]$ with $l_e' < l_e - l_\Phi - l_H - 3$ where l_Φ and l_H are security parameters (the first controlling statistical zero knowledge and the second being the size of the

challenge message in the PK protocol). Also the messages should lie in the small interval, i.e. m_i is limited by $\pm\{0,1\}^{lm\ -l\Phi-lH-2}$

CHAPTER 4. PRIME ENCODING METHOD

We now discuss how attributes are typically encoded in the CL credential system using Prime Encoding Method[1]. The CL credential system provides the means to efficiently encode a number of attributes into an anonymous credential. Each finite set attribute is represented by prime numbers. The product of the prime number is included in the CL credential system.

4.1. Setup

The three parties who are involved in the credential system are Issuer, User and Verifier:

Issuer: This party knows the secret key of the CL signature scheme, and this party has the authority to issue credentials to other parties.

User: User engages in the signature scheme protocol with the issuer and possesses a signature (e,A,v). As the attributes are included in the signature, the signature works as user credential and the user uses this credential to engage in proofs of knowledge with the verifier.

Verifier: This party uses proofs of knowledge to check the presence or absence of the attributes in the credential.

4.2. Attribute Encoding

We assume the CL credential system includes t finite set attributes. The attribute vector ($a_1 \ldots . a_t$) represents the finite set attributes, where $a_i \in (1 \ldots . n_i)$ shows different values of ith attribute set. Let $e_1 \ldots e_h$ be h distinct prime numbers where $h=\sum_{i=1}^{t} ni$. We assign each prime number to each element of the attribute and label them as $e_{(1,1)}, \ldots . e_{(1,n1) \ldots .} e_{(i,ni)}, \ldots \ldots e_{(t,ni).}$

The attributes are encoded into E as $E=\prod_{i=1}^{t} e(i, ai)$. After computing E, user gets a signature (e,A,v) on the messages m_0, E, where m_0 denotes the secret key of the user (CL) and E encodes all the finite set attributes.

4.3. Proofs about Attributes

For all the following proofs, user has to prove that

-He possess a valid credential on m_0, E

-He is able to prove the logical relations explained in the proofs

AND Relation:

Let us say that user wants show to the verifier that his credential contains one or more attributes without revealing the attributes. Let us assume that the list ($e_1 \ldots e_k$) denotes the encoding of the attributes. The user has to prove to the verifier that the list is present in the credential. To do this, the user shows to the verifier that $\prod_{j=1}^{k} ej|E$. To convince the verifier, he shows to the verifier that he knows μ' such that $E= (\prod_{j=1}^{k} ej)\mu'$ and runs the following with the verifier.

PK{(ε,v',μ_0,μ'):

$Z \Xi \pm R_0^{\mu 0} (R_1^{\Pi j=1\ ej})^{\mu'} A'^{\varepsilon} S^{v'} (\text{mod } n) \Lambda$

$\Lambda\ \mu_i \in \pm\{0,1\}^{lm} \Lambda\ \varepsilon \in [2^{le-1}+1, 2^{le}-1]\} \Lambda\ \mu' \in \pm\{0,1\}^{lm-lk}$

$l_{k=} \prod_{j=1}^{k} ej$

NOT Relation:

Here the user wants to prove to the verifier that the given list ($e_1 \ldots e_k$) is not present in the credential. In other words for $1 \leq j \leq k, e_j$ † E. This is equivalent to showing that that the product $\prod_{j=1}^{k} ej$ is relatively prime to E or the GCD ($\prod_{j=1}^{k} ej$,E)=1.To do this, the user convinces the verifier that he knows two integers, a and b, such that aE+b $\prod_{j=1}^{k} ej$=1. Note that a and b do not exist if GCD ($\prod_{j=1}^{k} ej$, E)>1 and they can be efficiently proved using a Euclidian algorithm [15].

The prover commits to an integer commitment to the value of E in $D=g^E h^r$ and sends D to the verifier and proves the knowledge of a and b with the following protocol.

PK{(ε,v',μ_0,μ ,α,β,ρ,ρ')}:

$Z \equiv \pm R_0^{\mu 0} R_1^{\mu} A'^{\varepsilon} S^{v'} \pmod n \; \Lambda$

$D \equiv \pm g^{\mu} h^{\rho} \pmod n \;\; \Lambda \;\; g \equiv \pm D^{\alpha} g^{(\Pi ej)\beta} h^{\rho'} \pmod n$

$\Lambda \; \mu_{0\ldots}\mu_l, \mu \in \pm\{0,1\}^{lm} \; \Lambda \; \varepsilon \in [2^{le-1}+1, 2^{le}-1]$

Note: α and β are representing a and b respectively

From $D \equiv \pm g^{\mu} h^{\rho} \pmod n$ and $g \equiv \pm D^{\alpha} g^{(\Pi ej)\beta} h^{\rho'} \pmod n$, we get $g \equiv \pm g^{\mu\alpha} g^{(\Pi ej)\beta} h^{\rho'} \pmod n$. Assuming the user doesn't know the value of $\log_g h$, the above statement shows that the user knows a α and β such that $1=\alpha\mu + \beta\prod_{j=1}^{k} ej$. It is obvious that α and β exist if and only if GCD $(\prod_{j=1}^{k} ej, E)=1$ and from the first proof the verifier is convinced that μ=E is present in the credential. Hence, the list ($e_1 \ldots e_k$) of attributes is not present in the credential.

OR Relation:

In the OR relation, the user wants to prove to the verifier that one of the attributes present in the list ($e_1 \ldots e_k$) is contained in the credential. For example, the user has to prove a statement such as "I'm a student, a retiree, or unemployed" as might be the case if one is eligible for a reduced entrance fee to a museum. More specifically, the "set membership" [14] proof is realized over encoded value. In order to do so, the user should be able to compute *a* such that $ae=\prod_{j=1}^{k} ej$. As we know that all the e_j's are prime, and if $e \notin (e_1 \ldots e_k)$ then such *a* does not exist. Also he has to show that the same e is present in E. To do this he has to prove the knowledge of *b* such that *eb*=E. The following proof is used to prove the OR relation:

PK{(ε,v',μ_0,μ ,α,β,ρ,ρ',δ,ρ'',ψ,ϒ,ϕ,σ,ξ)}:

$Z \equiv \pm R_0^{\mu 0} R_l^{\mu} A'^{\varepsilon} S^{v'} \pmod n \; \Lambda$

$D \equiv \pm g^{\alpha} h^{\rho} \pmod{n} \ \Lambda \ g^{\Pi ej} \equiv \pm D^{\delta} h^{\rho'} \pmod{n} \Lambda \ 1 \equiv \pm D^{\beta} g^{\mu} h^{\rho''} \pmod{n} \Lambda$

$\check{D} = \hat{g}^{\alpha} \hat{h}^{\phi} \Lambda \ \hat{g} = (\check{D}/\hat{g})^{\Upsilon} \hat{h}^{\psi} \Lambda \ \hat{g} = (\check{D}\hat{g})^{\sigma} \hat{h}^{\xi}$

$\Lambda \ \mu_{0 \ldots} \mu_{l}, \mu \in \pm\{0,1\}^{lm} \ \Lambda \ \varepsilon \in [2^{le-1} + 1, 2^{le} - 1]$

From $D \equiv \pm g^{\alpha} h^{\rho} \pmod{n} \ \Lambda \ g^{\Pi ej} \equiv \pm D^{\beta} h^{\rho'} \pmod{n}$, we get $g^{\Pi ej} \equiv \pm g^{\alpha\delta} h^{\rho'} h^{\delta\rho} \pmod{n}$. Assuming the user doesn't know $\log_g h$, these statements show that the user knows α and δ such that $\alpha\delta = \prod_{j=1}^{k} ej$, which means the user is committing to α, which divides the list of attributes. That is, the attribute *e* is present in the list.

$D \equiv \pm g^{\alpha} h^{\rho} \pmod{n}$ and $1 \equiv \pm D^{\beta} g^{\mu} h^{\rho''} \pmod{n}$ we get $1 \equiv \pm g^{\alpha\beta+\mu} h^{\rho''+\rho\beta} \pmod{n}$. Again by assuming the user doesn't know $\log_g h$, these statements show that the user knows α and β such that $-\alpha\beta = \mu$, which means the user is committing to α, which divides E. That is, the attribute *e* is present in the credential.

Now we have to see that α is not equal to ±1. As 1 divides both E and $\prod_{j=1}^{k} ej$. To do this we need an additional group of prime order q and two generators $\hat{g}$ and $\hat{h}$. $\check{D} = \hat{g}^{\alpha} \hat{h}^{\phi}$ and $\hat{g} = (\check{D}/\hat{g})^{\Upsilon} \hat{h}^{\psi}$ we get $\hat{g} = (\hat{g}^{\alpha}/\hat{g})^{\Upsilon} \hat{h}^{\psi}$. Assuming the hardness of computing $\log_{\hat{g}} \hat{h}$, the user knows α and Υ such that $1 \equiv \Upsilon(\alpha - 1) \bmod$ so α is not equal to 1. Similarly from $\check{D} = \hat{g}^{\alpha} \hat{h}^{\phi}$ and $\hat{g} = (\check{D}\hat{g})^{\sigma} \hat{h}^{\xi}$ we get $1 \equiv (\alpha + 1)\sigma \bmod q$. Hence, α is not equal to -1.

CHAPTER 5. PRIME MODULO ENCODING METHOD

Here we provide a new method of encoding attributes in CL signatures using the same setup as used in the prime encoding method but using the Chinese remainder theorem (CRT) in Prime Modulo Encoding Method.

5.1. Attribute Encoding

Suppose we want to encode t finite set or integer attributes into a CL credentials. Each of this set as n number of attributes. Each of this attribute are encoded as $a_1 \ldots a_n$ where $a_i \in (1 \ldots . n_i)$ shows different value of *i*th attribute set.For each i we pick a prime moduli e_i such that $e_i > n_i$.

Using the Chinese Remainder Theorem, we find an integer E, where $0 \leq E \leq \Pi_{i=1}^{t} e_i$, using e_i's as moduli and a_i's as remainders such that $E \equiv a_i \pmod{e_i}$,where a_i is the encoding of the attribute corresponding to the ith attribute set.

5.2. Proofs about Attributes using Prime Modulo Encoding Method

In this section, we assume all the binary and finite set attributes are encoded into one message, namely m_o and E, where E represents the encoding of the attributes and m_0 includes the user's secret key.

AND Relation:

Here the user wants to show to the verifier that the subset $\{(a_{1'}, e_{1'}) \ldots . (a_{k'}, e_{k'})\}$ of the attribute vector $\{(a_1, e_1) \ldots . (a_t, e_t)\}$ is present in the credential E. Since the user has to show that all k attributes are encoded both in E and E', the user forms an integer E' such that $E' \equiv a_{i'} \pmod{e_{i'}}$ and $1 \leq i \leq k$.E' can be found out by using the Chinese remainder theorem and E is the solution of $E \bmod e_1 = a_1$; $E \bmod e_2 = a_2 \ldots . E \bmod e_t = a_t$. Hence, using the congruence property of the Chinese remainder theorem [18] we know $\prod_{j=1}^{k} ej$ divides E-E'. In order to show this as E is not known to the verifier, the user can prove the knowledge of an integer P such that $E\text{-}E' = P * \prod_{j=1}^{k} ej$ or

$E = E' + P.\Pi_{i=1}^{k} e_i$.

To convince the verifier that he was issued a credential that contains subset $\{(a_{1'},e_{1'})\ldots.(a_{k'},e_{k'})\}$ of the attribute vector $\{(a_1,e_1)\ldots.(a_t, e_t)\}$, the user engages in the following proof with the verifier:

$PK\{(\varepsilon,v',\mu_0,\pi)\}$:

$Z \equiv \pm R_0^{\mu 0} R_1^{E'} (R_1^{\Pi ei})^{\pi} A'^{e} S^{v'} (\text{mod } n)\ \Lambda$

$\Lambda\ \mu_0 \epsilon \pm \{0,1\}^{lm}\ \Lambda\ \pi\ \pm \{0,1\}^{lm-lk}$

$\Lambda\ \varepsilon \epsilon [2^{le-1}+1, 2^{le}-1]$

Where l_k is the size of $\prod_{i=1}^{k} ei$ in bits.

Note: As E' is known both to the user and verifier $R_1^{E'}$ can be computed.

We require that the size of the message input into the signature scheme be limited: $\mu_i \epsilon \{0,1\}^{lm-l\infty-lH-2}$; also, we require that ε should lie in a small interval $[2^{le-1}-2^{le-l\infty-lH-4}, 2^{le-1}-2^{le-l\infty-lH-4}]$.

<u>Protocol in detail</u>: $Z \equiv \pm R_0^{\mu 0} R_1^{E'} (R_1^{\Pi ei})^{\pi} A'^{\varepsilon} S^{v'} (\text{mod } n)$

Common inputs: z,R_0,R_1,A',S,N with parameters lm,l∞,le. Public keys g,h

Prover's inputs: μ0, π, ε, v', A, μ_0 is of length $[-2^{lm-l\infty-lH-2}, 2^{lm-l\infty-lH-2}]$, π is of length $[-2^{lm-l\infty-lH-lk-2}, 2^{lm-l\infty-lH-lk-2}]$ and ε is of length $[2^{le-1}-2^{le-l\infty-lH-4}, 2^{le-1}+2^{le-l\infty-lH-4}]$

A',ε,v' is a valid signature.

prover→verifier: $\mu_{0'} \epsilon_r \{[-2^{lm-2}\ldots\ldots 2^{lm--2}\}$,$\pi' \epsilon_r \{[-2^{lm-lk-2}\ldots\ldots 2^{lm-lk--2}\}$, $\epsilon' \epsilon_R \{-2^{le-4}$ ……$2^{le-4}\}$ computes $Z' = R_0^{\mu 0'} R_1^{\pi'} A'^{\varepsilon'} S^{v1} (\text{mod } n)$ and sends Z' to the verifier.

Prover ←verifier: The verifier chooses uniformly at random the challenge c from $\{0,1\}^{lH}$ and sends it to the prover.

Prover→ verifier: The prover calculates

$S1 = \mu_{0'} - c\mu_0$

$S2 = \pi' - c\pi$

$S3 = \epsilon' - c(\varepsilon - 2^{le-1})$

$S4 = v^1 - cv'$

Sends S1, S2, S3, and S4 to the verifier. The verifier checks. $-2^{lm-1} < S1 < 2^{lm-1}$

$-2^{lm-lk-1} < S2 < 2^{lm-lk-1}$

$-2^{le-3} < S3 < 2^{le-3}$

$R_0^{s1} R_1^{s2} A'^{s3-c\{(2\wedge le)-1\}} S^{s4} z^c = Z'$

PROOF:

Completeness:

If the prover and the verifier act as described in the AND relation, then

$R_0^{s1} R_1^{s2} A'^{s3-c(2le-1)} S^{s4}$

$= (R_0)^{\mu o'-c\mu o} (R_1)^{\Pi'-c\Pi} (A')^{\epsilon'-c\{\epsilon-2\wedge(le)-1\}-c\{(2\wedge le)-1\}} S^{v1-cv'}$

$= (R_0)^{\mu o'} (R_1)^{\Pi'} (A')^{\epsilon'} S^{v1} Z^{-c}$

$= Z' Z^{-c}$

Soundness:

$R_0^{s1} R_1^{s2} A'^{s3-c\{(2\wedge le)-1\}} S^{s4} = Z' Z^{-c}$

$R_0^{s1'} R_1^{s2'} A'^{s3'-c'\{(2\wedge le)-1\}} S^{s4'} = Z' Z^{-c'}$

$R_0^{s1-s1'} R_1^{s2-s2'} A'^{s3-c\{(2\wedge le)-1\}-s3'+c'\{(2le)-1\}} S^{s4-s4'} = Z^{-c+c'}$

$R_0^{s1-s1'/c'-c} R_1^{s2-s2'/c'-c} A'^{s3-c\{(2le)-1\}-s3'+c'\{(2\wedge le)-1\}/c'-c} S^{s4-s4'/c'-c} = Z$

Proof of Knowledge: The knowledge extractor K^{p*} works as follows:

1. Run K^{P*} and receive message Z from P^*
2. A c=0 is sent to P^* and we received s_1^0, s_2^0, s_3^0, s_4^0
3. We rewind P^* before step 2, sent c=1 and received s_1^1, s_2^1, s_3^1, s_4^1

4. Output s_1^1- s_1^0, s_2^1-s_2^0, s_3^1-s_3^0, s_4^1- s_4^0

If s_1^1,s_1^0, s_2^1,s_2^0, s_3^1,s_3^0, s_4^1, s_4^0 are all correct answers to the respective challenges then the K^{p*} output correct witnesses.

Here we have $R_1^E \equiv R_1^{E'} (R_1^{\Pi ei})^{\pi}$ from which we can conclude $E \equiv E' + \prod_{j=1}^{k} ej$ where E' can be computed both by the verifier and the user. Therefore, we can say that the subset of the attribute vector is present in E.

NOT Relation:

Here the user wants to show that his credential does not contain the subset $\{(a_{1'},e_{1'})\ldots.(a_{k'},e_{k'})\}$ of the attribute vector $\{(a_1,e_1)\ldots.(a_t,e_t)\}$. To prove this the user will find an E' using the Chinese remainder theorem. E' encodes all the attributes $\{(a_{1'},e_{1'})\ldots.(a_{k'},e_{k'})\}$ such that for all $1 \leq i \leq k$, $E' \equiv a_{i'} (\text{mod } e_{i'})$ where e_i is a prime number corresponding to the *i*th attribute set of the vector.

Furthermore, the attributes of the user are encoded in E and the user has to show that $E \neq E' (\text{mode}_{i'})$. In other words, it means that $e_{i'} \nmid E\text{-}E'$. This is equivalent to showing that the products of e_i's are coprime with E-E', or GCD (E-E', $\prod_{i=1}^{k} ei'$) = 1. Then according to the extended Euclidian algorithm, there exist two integers, a and b, such that $a(E\text{-}E')+b(\prod_{i=1}^{k} ei')=1$. Note: a and b do not exist if the GCD (E-E', $\prod_{i=1}^{k} ei$) > 1.

The user commits to the integer commitment to E by sending $D=g^E h^r$ to the verifier and then engages with the following proof with the verifier, where α and β represent the integers a and b.

PK $\{(\varepsilon, v', \mu_0, \mu_1, \alpha, \beta, \rho, \rho')\}$:

$Z \equiv \pm R_0^{\mu 0} R_1^{\mu 1} A'^{\varepsilon} S^{v'} (\text{mod } n)$ Λ

$D \equiv \pm g^{\mu 1} h^{\rho} (\text{mod} n)$ Λ $g \equiv \pm D^{\alpha} g^{(-E')\alpha} g^{(\Pi ei')\beta} h^{\rho'} (\text{mod} n)$

$\Lambda\ \mu_{0,}\ \mu_1\epsilon \pm \{0,1\}^{lm}$

$\Lambda\ \varepsilon\ \epsilon\ [2^{le-1}+1,\ 2^{le}-1]$

Protocol in detail:

Common inputs: z,R_0,R_1,A',S,D with parameters lm,l∞,le .Public keys g,h

Prover's inputs:μ_0,μ_1,ε,α,β,ρ,ρ',v',μ_0 ,μ_1 is of length $[-2^{lm-l\infty-lH-2},\ 2^{lm-l\infty-lH-2}]$ and ε is of length $[2^{le-1}-2^{le-l\infty-lH-4},\ 2^{le-1}+2^{le-l\infty-lH-4}]$

A',ε,v' is a valid signature.

prover→verifier: The prover then chooses uniformly at random $v^1,\alpha^1,\beta^1,\rho^1,\rho^2$, μ0',μ1' ϵ_r $\{[-2^{lm-2}\ldots\ldots 2^{lm--2}\}$,ϵ' ϵR $\{-2^{le-4}\ldots\ldots\ldots\ldots 2^{le-4}\}$ compute $D' \equiv \pm g^{\mu 1}\ h^{\rho 1}$ and $Z' \equiv$

$R_0^{\mu 0'} R_1^{\mu 1'} A'^{\varepsilon'} S^{v1} (\text{mod } n)$,

$g' \equiv \pm D'^{\alpha 1}\ g^{(-E')\alpha 1} (g^{\Pi ei'})^{\beta 1} h^{\rho 2}$ (modn) and sends D', Z', g' to the verifier.

Prover ←verifier: The verifier chooses uniformly at random the challenge c from $\{0,1\}^{lH}$ and sends it to the prover.

Prover→ verifier: The prover calculates

$S1 = \mu_{0'} - c\mu_0$

$S2 = \mu_{1'} - c\mu_1$

$S3 = \epsilon' - c(e-2^{le-1})$

$S4 = v^1 - cv'$

$S5 = \alpha^1 - c\alpha$

$S6 = \beta^1 - c\beta$

$S7 = \rho^1 - c\rho$

$S8 = \rho^2 - c\rho'$ and sends S1, S2, S3, S4, S5, S6, S7, S8 to the verifier. The verifier checks

$$-2^{lm-1} < S1, S2 < 2^{lm-1}$$

$$-2^{le-3} < S3 < 2^{le-3}$$

$$\pm R_0^{\,s1} R_1^{s2} A'^{s3-c\{(2\wedge le)-1)\}} S^{s4} z^{c} \equiv Z'$$

$$\pm g^{\,s2} h^{s7} D^{c} \equiv D'$$

$$\pm D'^{s5}\, g^{(-E')s5} (g^{\Pi ei'})^{s6} h^{s8}\, g^{c} \equiv g'$$

PROOF:

Completeness: If the prover and the verifier act as described in the NOT relation, then

$R_0^{\,s1} R_1^{s2} A'^{s3-c\{(2\wedge le)-1\}} S^{s4}$

$= (R_0)^{\mu o'-c\mu o} (R_1)^{\Pi'-c\Pi} (A')^{\epsilon'-c\{\epsilon-(2\wedge le)-1\}-c\{(2\wedge le)-1\}} S^{v1-cv'}$

$= (R_0)^{\mu o'} (R_1)^{\Pi'} (A')^{\epsilon'} S^{v1} Z^{-c}$

$= Z'Z^{-c}$

$g^{\,s2} h^{s7}$

$= g^{\mu 1'-c\mu 1} h^{\rho'-c\rho}$

$= g^{\mu 1'} h^{\beta 1} D^{-c}$

$= D'D^{-c}$

$D'^{s5}\, g^{(-E')s5} (g^{\Pi ei})^{s6} h^{s8}$

$= (g^{\mu 1} h^{\rho 1})^{\alpha 1-c\alpha} g^{(-E')\alpha 1-c\alpha} (g^{\Pi ei})^{\beta 1-c\beta} h^{\rho 2-c\rho'}$

$= (g^{\mu 1-E'})^{\alpha 1} (g^{\Pi ei})^{\beta 1} h^{\rho 1\alpha 1} h^{\rho 2} (g^{\mu 1-E'})^{-c\alpha} (g^{\Pi ei})^{-c\beta} h^{-cp1\alpha} h^{-c\rho'}$

$= g'g^{-c}$

Soundness:

$R_0^{\,s1} R_1^{s2} A'^{s3-c\{2\wedge(le)-1\}} S^{s4} = Z'Z^{-c}$

$R_0^{\,s1'} R_1^{s2'} A'^{s3'-c'\{2\wedge(le-1)\}} S^{s4'} = Z'Z^{-c'}$

$R_0^{\,s1-s1'} R_1^{s2-s2'} A'^{s3-c'\{(2\wedge le)-1\}-s3'+c'\{(2\wedge le)-1\}} S^{s4-s4'} = Z^{-c+c'}$

$R_0^{\,s1-s1'/c'-c} R_1^{s2-s2'/c'-c} A'^{s3-c\,\{(2\wedge le)-1\}-s3'+c'\{(2\wedge le)-1\}/c'-c} S^{s4-s4'/c'-c} = Z$

$g^{s2}\, h^{s7} = D'\ D^{-c}$

$g^{s2'}\, h^{s7'} = D'\ D^{-c'}$

$g^{s2-\ s2'}\, h^{\ s7-s7'} = D^{-c+c'}$

$g^{s2-\ s2'/c'-c}\, h^{\ s7-s7'/c'-c} = D$

$D'^{s5}\ g^{(-E')s5}\ (g^{\Pi ei'})^{s6} h^{s8} \equiv g'\, g^{-c}$

$D'^{s5'}\ g^{(-E')s5'}\ (g^{\Pi ei'})^{s6'} h^{s8'} \equiv g'\, g^{-c'}$

$D'^{s5-s5'}\ g^{(-E')s5-s5'}\ (g^{\Pi ei'})^{s6-s6'} h^{s8-s8'} \equiv g^{-c+c'}$

$D'^{s5-s5'/-c+c'}\ g^{(-E')s5-s5'/-c+c'}\ (g^{\Pi nj})^{s6-s6'/-c+c'} h^{s8-s8'/-c+c'} \equiv g$

Proof of Knowledge: The knowledge extractor K^{p*} works as follows:

1. Run K^{p*} and receive message Z,D from P^*
2. A c=0 is sent to P^* and we received $s_1^0, s_2^0, s_3^0, s_4^0, s_5^0, s_6^0, s_7^0, s_8^0$
3. We rewind P^* before step 2, sent c=1 and received $s_1^1, s_2^1, s_3^1, s_4^1, s_5^1, s_6^1, s_7^1, s_8^1$
4. Output $s_1^1 - s_1^0, s_2^1 - s_2^0, s_3^1 - s_3^0, s_4^1 - s_4^0, s_5^1 - s_5^0, s_6^1 - s_6^0, s_7^1 - s_7^0, s_8^1 - s_8^0$

If $s_1^1, s_1^0, s_2^1, s_2^0, s_3^1, s_3^0, s_4^1, s_4^0, s_5^1, s_5^0, s_6^1, s_6^0, s_7^1, s_7^0, s_8^1, s_8^0$ are all correct answers to the respective challenges then the K^{p*} output correct witnesses.

As both the user and the verifier can compute $(g^{-E'})$ from $D \equiv \pm g^{\mu 1} h^{\rho}$ (modn) and $g \equiv \pm D^{\alpha}\ g^{(-E')\alpha}\, g^{(\ \Pi ei'\ )\beta} h^{\rho'}$ (modn) , the user is proving the knowledge of α and β such that $g \equiv \pm g^{\alpha\mu 1}\ g^{(-E')\alpha}\, g^{(\ \Pi ei'\ )\beta} h^{\rho'}\, h^{\alpha\rho}$. Assuming the hardness of computing $\log_g h$, the statements $D \equiv \pm g^{\mu 1} h^{\rho}$ (modn) and $g \equiv \pm D^{\alpha}\ g^{(-E')\alpha}\, g^{(\ \Pi ei'\ )\beta} h^{\rho'}$ (modn) are enough to show that $\alpha(E-E') + \beta(\Pi_{i=1}^{k} e_i) = 1$ which implies $GCD(E-E', \prod_{j=1}^{k} ei') = 1$, and it means none of the attributes of the vector $\{(a_1,e_1)\ldots(a_k,e_k)\}$ are present in E.

OR Relation:

Let us now show how we implement the existence of one of the attribute vectors in the credential without revealing the attribute to the verifier. We assume we are given the encodings of the attribute vector $\{(a_{1'},e_{1'})\ldots.(a_{k'},e_{k'})\}$ belonging to different attribute sets. We encode all the attributes in E'. The user has to show that e ϵ $\{e_{1'},e_{2'}\ldots.e_{k'}\}$ is present both in E and E'. The idea we use here is that if the credential contains e that is present in E', then e | E-E' or there exists an s.t *a*e =E-E'. If e is not in the credential, then no such *a* exists. Also the user has to show that the same e is present in $\{(a_{1'},e_{1'})\ldots.(a_{k'},e_{k'})\}$. To do this, the user has to show that there exists a *b* such that *b*e $=\prod_{j=1}^{k} ei'$. If e is not present in $\{e_{1'}, e_{2'}\ldots.e_{k'}\}$, then no such b exists.

To prove that his credential contains one of the attribute's values, the user employs the following protocol: First the user computes a commitment D to the attribute present in her credential (in the same way as for the other protocols), sends it to the verifier and then runs the following protocol with the verifier.

However we must see that the commitment does not contain ±1, as 1 divides both E and $\prod_{j=1}^{k} ei'$. To do this we need an additional group of prime order q and two generators ĝ and ĥ. The user also computes the commitment Ď and runs the following protocol with the verifier:

PK$\{(\varepsilon,v',\mu_0,\mu_1,\alpha,\beta,\rho,\rho',\delta,\rho'',\psi,\Upsilon,\phi,\sigma,\xi)\}$:

$Z\equiv\pm R_0^{\mu 0}R_1^{\mu 1}A^{'\varepsilon}S^{v'} \pmod n \ \Lambda$

$D\equiv \pm g^{\delta}h^{\rho} \pmod n \ \Lambda \ g^{\Pi ei'}\equiv \pm D^{\beta}h^{\rho'} \pmod n \Lambda \ 1\equiv\pm D^{\alpha}g^{\mu 1}g^{-E'}h^{\rho''} \pmod n \ \Lambda \ \check{D}=\hat{g}^{\delta}\hat{h}^{\phi} \ \Lambda$

$\hat{g}=(\check{D}/\hat{g})^{\Upsilon}\hat{h}^{\psi}$

$\Lambda \ \hat{g}=(\check{D}\hat{g})^{\sigma}\hat{h}^{\xi}$

$\Lambda \ \mu_0,\mu \epsilon \pm\{0,1\}^{lm} \ \Lambda \ \varepsilon \epsilon [2^{le-1}+1, 2^{le}-1]$

Note: Here δ, β and α represents e,*b* and *a* respectively.

Protocol in detail:

Common inputs: z,R_0,R_1,A',S,D with parameters lm,l∞,le .Public keys g,h

Prover's inputs: μ_0,μ_1,ε,α,β,ρ,ρ',v',μ_0 ,μ_1 is of length $[-2^{lm-l\infty-lH-2}, 2^{lm-l\infty-lH-2}]$ and ε is of length $[2^{le-1}-2^{le-l\infty-lH-4}, 2^{le-1}+2^{le-l\infty-lH-4}]$

A',ε,v' is a valid signature.

prover→verifier: The prover then chooses uniformly at random v^1,α^1,β^1,ρ^1,ρ^2, μ0',μ1' , ϵ_r $\{[-2^{lm-2}\ldots\ldots 2^{lm--2}\}$,ϵ' ϵR $\{-2^{le-4}\ldots\ldots\ldots\ldots 2^{le-4}\}$ computes $Z' = R_0^{\mu 0'} R_1^{\mu 1'} A'^{\varepsilon'} S^{v1} (\text{mod } n)$, $D' \equiv g^{\delta 1} h^{\rho 1} (\text{modn})$, $g'^{\Pi ei'} \equiv D^{\beta 1} h^{\rho 2} (\text{modn})$, $g'' \equiv D^{\alpha 1} g^{\mu 1} g^{-E'} h^{\rho 3} (\text{modn})$ $\check{D}' = \hat{g}^{\delta 1} \hat{h}^{\phi'}$ Λ $\hat{g}' = (\check{D}/\hat{g})^{\Upsilon''} \hat{h}^{\psi'}$ Λ $\hat{g} = (\check{D}\hat{g})^{\sigma'} \hat{h}^{\xi'}$ and sends D', Z', g', Ď' to the verifier.

Prover ←verifier: The verifier chooses uniformly at random the challenge c from $\{0,1\}^{lH}$ and sends it to the prover.

Prover→ verifier: The prover calculates

S1= $\mu_{0'} - c\mu_0$

S2= $\mu_{1'} - c\mu_1$

S3 = ϵ' -c(e-2^{le-1})

S4 = v^1 –cv'

S5 =α^1-cα

S6 = β^1-cβ

S7 =ρ^1 –cρ

S8=ρ^2-cρ'

S9=ρ^3-cρ''

S10=φ'-cφ

S11=ɣ’-cɣ

S12=ψ’-cψ

S13=ϭ’-cϭ

S14=ξ’-cξ

S15=δ1-cδ

and sends S1, S2, S3, S4, S5, S6, S7, S8,S9,S10,S11,S12,S13,S14 and S15 to the verifier. The verifier checks

$$-2^{lm-1} < S1,S2 < 2^{lm-1}$$

$$-2^{le-3} < S3 < 2^{le-3}$$

$$R_0^{s1} R_1^{s2} A'^{s3-c\{2\wedge(le)-1} S^{s4} z^c \equiv Z'$$

$$g^{S15} h^{S7} D^c \equiv D'$$

$$D^{s6} h^{s8} (g^{\Pi ej})^c = (g'^{\Pi ej})$$

$$D^{s5} g^{s1} g^{-E'} h^{s9} = g''$$

$$\hat{g}^{s5} \hat{h}^{s10} \check{D}^c = \check{D}'$$

$$(\check{D}/\hat{g})^{s11} \hat{h}^{s12} \hat{g}^c = \hat{g}'$$

$$(\check{D}\hat{g})^{s13} \hat{h}^{14} \hat{g}^c = \hat{g}'$$

PROOF:

Completeness: If the prover and the verifier act as described in the OR relation, then

$$R_0^{s1} R_1^{s2} A'^{s3-c\{(2\wedge le)-1\}} S^{s4}$$

$$= (R_0)^{\mu o'-c\mu o} (R_1)^{\Pi'-c\Pi} (A')^{\epsilon'-c(\epsilon-2le-1)-c(2le-1)} S^{v1-cv'}$$

$$= (R_0)^{\mu o'} (R_1)^{\Pi'} (A')^{\epsilon'} S^{v1} Z^{-c}$$

$$g^{S15} h^{S7}$$

$$= g^{\delta 1-c\delta} h^{\rho 1-c\rho}$$

$= g^{\delta 1}\, h^{\rho 1}\, (g^{\delta}\, h^{\rho})^{-c}$

$= D' D^{-c}$

$D^{s6} h^{s8}$

$= D^{\beta 1 - c\beta}\, h^{\rho 2 - c\rho}$

$= (g^{'\Pi ej})\, (g^{\Pi ej})^{-c}$

$D^{s5}\, g^{s1}\, g^{-E'}\, h^{s9}$

$= D^{\alpha 1 - c\alpha}\, g^{\mu 1' - c\mu 1}\, g^{-E'}\, h^{\rho 3 - c\rho''}$

$= D^{\alpha 1}\, g^{\mu 1'}\, g^{-E'}\, h^{\rho 3}\, (D^{\alpha}\, g^{\mu 1}\, g^{-E'}\, h^{\rho''})^{-c}$

$= g^{,,}(1)^{-c}$

Soundness:

$R_0^{\,s1}\, R_1^{s2} A^{'s3 - c2(le-1)} S^{s4} = Z' Z^{-c}$

$R_0^{\,s1'}\, R_1^{s2'} A^{'s3' - c'2(le-1)} S^{s4'} = Z' Z^{-c'}$

$R_0^{\,s1 - s1'}\, R_1^{s2 - s2'}\, A^{'s3 - c\{(2\hat{}le) - 1\} - s3' + c'\{(2\hat{}le) - 1\}} S^{s4 - s4'} = Z^{-c+c'}$

$R_0^{\,s1 - s1'/c' - c}\, R_1^{s2 - s2'/c' - c}\, A^{'s3 - c\{(2\hat{}le) - 1\} - s3' + c'\{(2\hat{}le) - 1)/c' - c} S^{s4 - s4'/c' - c} = Z$

$g^{S15}\, h^{S7} = D' D^{-c}$

$g^{S15'}\, h^{S7'} = D' D^{-c'}$

$g^{S15 - s15'}\, h^{S7 - s7'} = D^{-c+c'}$

$g^{S15 - s15'/c' - c}\, h^{S7 - s7'/c' - c} = D$

$D^{s6} h^{s8} = (g^{'\Pi ei'})\, (g^{\Pi ei'})^{-c}$

$D^{s6'} h^{s8'} = (g^{'\Pi ei'})\, (g^{\Pi ei'})^{-c'}$

$D^{s6 - s6'} h^{s8 - s8'} = (g^{\Pi ei'})^{-c+c'}$

$D^{s6 - s6'/c' - c} h^{s8 - s8'/c' - c} = (g^{\Pi ei'})$

$D^{s5}\, g^{s1}\, g^{-E'}\, h^{s9} = g''$

$D^{s5'} g^{s1'} g^{-E'} h^{s9'} = g''$

$D^{s5-s5'} g^{s1-s1'} h^{s9-s9'} = 1$

$D^{s5-s5'/c'-c} g^{s1-s1'/c'-c} h^{s9-s9'/c'-c} = 1$

Proof of Knowledge: The knowledge extractor K^{P^*} works as follows:

1. Run K^{P^*} and receive message Z,D, $g^{\Pi ei'}$, $\check{D}$ from P^*

2. A c=0 is sent to P^* and we receive s_1^0, s_2^0 , s_3^0, s_4^0, s_5^0, s_6^0, s_7^0, s_8^0, s_9^0, s_{10}^0, s_{11}^0, s_{12}^0, s_{13}^0, s_{14}^0 and s_{15}^0

3. We rewind P^* before step 2, sent c=1 and received s_1^1, s_2^1, s_3^1, s_4^1, s_5^1, s_6^1, s_7^1, s_8^1, s_9^1, s_{10}^1, s_{11}^1, s_{12}^1, s_{13}^1, s_{14}^1 and s_{15}^1

4. Output $s_1^1 - s_1^0$, $s_2^1 - s_2^0$, $s_3^1 - s_3^0$, $s_4^1 - s_4^0$, $s_5^1 - s_5^0$, $s_6^1 - s_6^0$, $s_7^1 - s_7^0$, $s_8^1 - s_8^0$, $s_9^1 - s_9^0$, $s_{10}^1 - s_{10}^0$, $s_{11}^1 - s_{11}^0$, $s_{12}^1 - s_{12}^0$, $s_{13}^1 - s_{13}^0$, $s_{14}^1 - s_{14}^0$ and $s_{15}^1 - s_{15}^0$.

If s_1^1, s_1^0, s_2^1, s_2^0, s_3^1, s_3^0, s_4^1, s_4^0, s_5^1, s_5^0, s_6^1, s_6^0, s_7^1, s_7^0, s_8^1, s_8^0 , s_9^1, s_9^0 , s_{10}^1, s_{10}^0 , s_{11}^1, s_{11}^0, s_{12}^1, s_{12}^0, s_{13}^1, s_{13}^0, s_{14}^1, s_{14}^0 and s_{15}^1, s_{15}^0 are all correct answers to the respective challenges, then the K^{P^*} output correct witnesses.

First from $D \equiv \pm g^{\delta} h^{\rho} \pmod{n}$ Λ $g^{\Pi ei'} \equiv \pm D^{\beta} h^{\rho'} \pmod{n}$,we get $g^{\Pi ei'} \equiv \pm g^{\beta\delta} h^{\rho'} h^{\delta\rho} \pmod{n}$. Assuming the user doesn't know $\log_g h$, these statements show that the user knows β and δ such that $\beta\delta = \prod_{j=1}^{k} ei'$ and this implies that $\delta | \prod_{j=1}^{k} ei'$, which means δ is one of the e*i*'s. Second, $D \equiv \pm g^{\delta} h^{\rho} \pmod{n}$ and $1 \equiv \pm D^{\alpha} g^{\mu 1} g^{-E'} h^{\rho''} \pmod{n}$ we get $1 \equiv \pm g^{\alpha\delta + \mu 1 - E'} h^{\rho'' + \rho\beta} \pmod{n}$. Again by assuming the user doesn't know $\log_g h$, these statements show that the user knows δ and α such that $\alpha\delta = \mu 1 - E'$, which means that δ is present in the credential; therefore, δ is present both in E and E'.

Now we have to see that α is not equal to ± 1, as the above two statements hold for ± 1. To do this, first we consider $\check{D} = \hat{g}^{\delta} \hat{h}^{\phi}$ Λ $\hat{g} = (\check{D}/\hat{g})^{\Upsilon} \hat{h}^{\Psi}$. As $\log_g h$ is difficult to compute we have

$1 \equiv \gamma(\delta - 1)$ from which we can say $\delta \ddagger 1$. Similarly from $\check{D} = \hat{g}^{\delta}\,\hat{h}^{\phi}$ and $\hat{g} = (\check{D}\hat{g})^{\sigma}\,\hat{h}^{\xi}$. As $\log_g h$ is hard to compute we get $1 \equiv (\delta + 1)\mathcal{O}$ from which δ is not equal to -1. Hence δ is present both in E and E' and also it is ± 1.

CHAPTER 6. COMPARISONS BETWEEN PRIME AND PRIME MODULO ENCODING

In this section we investigate the number of bits required to encode binary as well as finite set attributes both for the prime and prime modulo methods. We plot the graphs to show our comparisons. We claim that with the Prime Modulo Encoding method we can encode more attributes in 256-bit messages than with the prime encoding method. We compare the number of attributes needed to encode 256 bits of binary attributes as well as the number of bits required to encode five finite attributes

6.1. Binary Set Attributes

Let us assume that we want to encode *t* binary attributes. Each attribute α_i ($1 \leq i \leq t$) can take two values, either 0 or 1. In order to encode the binary attributes by the Camenisch-Groβ prime encoding method, we need 2*t* prime numbers. They are $e_{1,\ldots\ldots}e_{2t}$. In Prime Modulo Encoding method, to encode the attributes we need *t* prime numbers. They are $e_{1'},\ldots\ldots e_{t'}$. In order to minimize the size of E, we start with the smallest prime number, i.e. $e_1 = e_{1'} = 2$. So the number of bits needed to encode binary attributes through the Prime Modulo Encoding method is much less than that by Camenisch –Groβ. We plot a graph to compare the number of bits required to encode binary attributes starting with $e_1 = e_{1'} = 2$ and extend it to show how many attributes 256 bits can incorporate. The graph shows that almost 35 attributes for the Prime Modulo Encoding and around 43 attributes for the prime modulo encoding method can be incorporated in 256 bits.

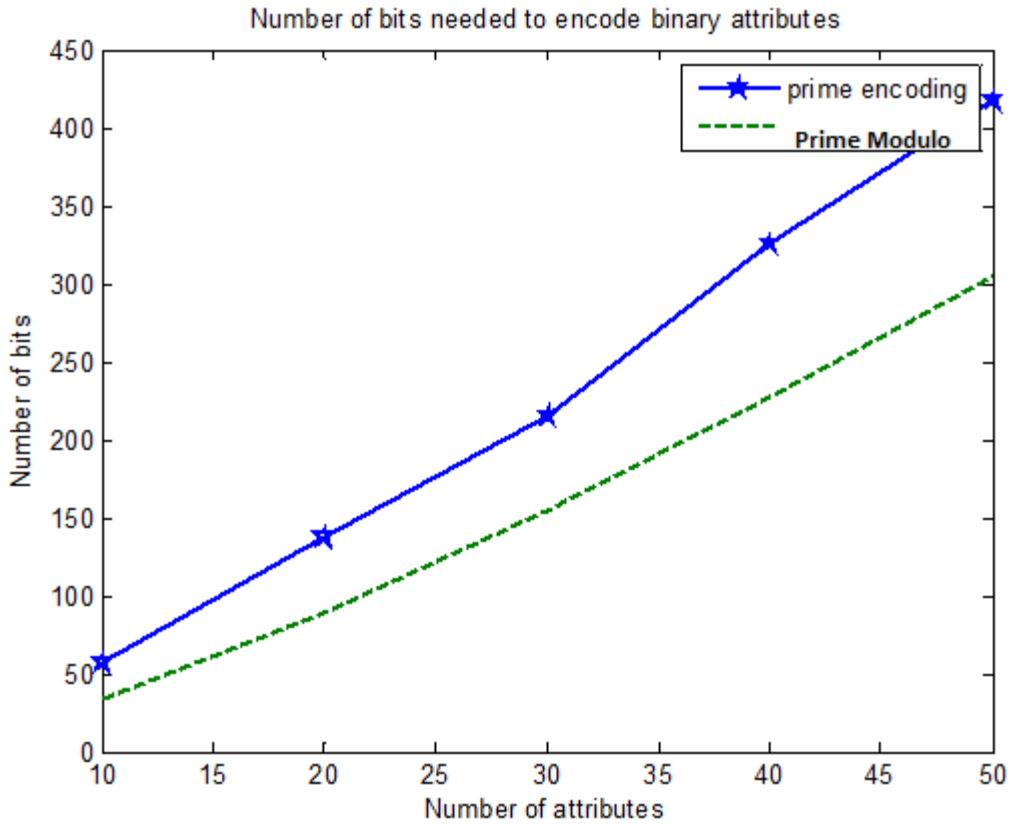

Figure 1. Number of bits required to encode binary attributes

6.2. Finite Set Attributes

Let us assume that we want to encode t attributes $\alpha_{1,\ldots.}\ \alpha_t$ where $\alpha_i \in \{0,\ldots.,t\text{-}1\}$ for $i \in \{1\ldots t\}$. In the prime encoding method we need t^2 prime numbers while in Prime Modulo Encoding method we need t prime numbers, all greater than or equal to t. We plot a graph starting with $e_1 = e_{1'} = 2$. We can see that the number of bits required to plot four attributes in the prime encoding method is around 22 bits, whereas for the Prime Modulo Encoding method it is only around 13 bits.

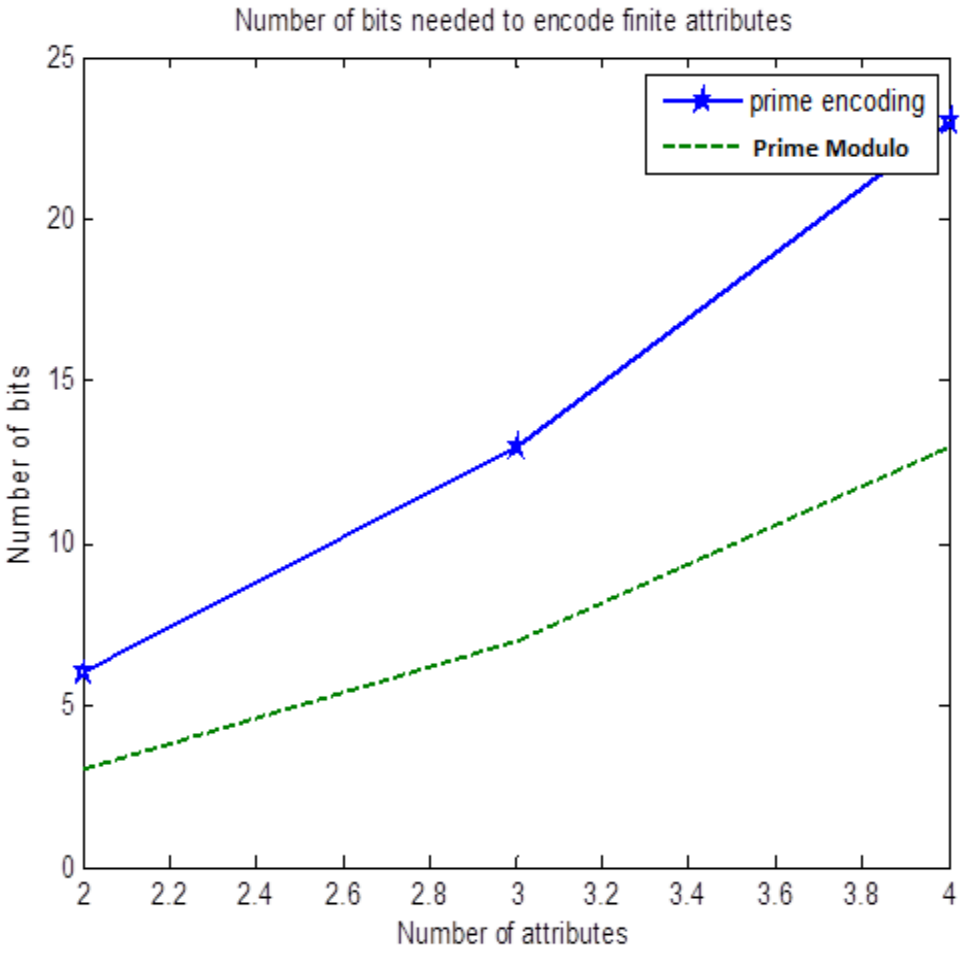

Figure 2. Number of bits required to encode finite set attributes

7. CONCLUSION

From the plots we can surely say that our Prime Modulo Encoding method incorporates more attributes than does the Camenisch-Groβ encoding method. Prime Modulo Encoding method facilitates proofs of possession, equality, as well as AND, NOT and OR proofs very efficiently. The new method also overcomes the fundamental limitation of all credential systems, namely their complexity is linear in total number of attributes, and it allows us to incorporate many finite-set attributes into a single attribute base and therefore boosts the performance of all proofs of possession. The Prime Modulo method used Chinese Remainder Theorem to encode the attributes. Prime Modulo Encoding requires no additional cryptographic assumptions apart from strong RSA, it targets the major attribute classes of credential systems, and it can be used in real applications, such as electronic identity cards and complex forms of professional and medical credentials. Further research should focus on string attributes though minority of the attributes requires string attributes like name and birthdays.

8. REFERENCES

[1] Camenisch, Jan, and Thomas Groß. "Efficient attributes for anonymous credentials." *Proceedings of the 15th ACM conference on computer and communications security*. ACM, 2008.

[2] Camenisch, Jan, and Anna Lysyankaya. "A signature scheme with efficient protocols." *Security in communication networks*. Springer Berlin Heidelberg, 2003. 268-289.

[3] Camenisch, Jan, and Anna Lysyankaya. "An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *"Advances in Cryptology—EUROCRYPT 2001*. Springer Berlin Heidelberg, 2001. 93-118.

[4] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, Proc. CRYPTO '97, volume 1294 of LNCS, pages 16–30. Springer-Verlag, 1997

[5] Damgård, Ivan. "Commitment schemes and zero-knowledge protocols. *"Lectures on Data Security*. Springer Berlin Heidelberg, 1999. 63-86.

[6] Damgård, Ivan, and Eiichiro Fujisaki. "A statistically-hiding integer commitment scheme based on groups with hidden order." *Advances in Cryptology—ASIACRYPT 2002*. Springer Berlin Heidelberg, 2002. 125-142.

[7] Chaum, David, et al. "Demonstrating possession of a discrete logarithm without revealing it." *Advances in Cryptology—CRYPTO'86*. Springer Berlin Heidelberg, 1987.

[8] Schnorr, Claus-Peter. "Efficient signature generation by smart cards." *Journal of Cryptology* 4.3 (1991): 161-174.

[9] Brands, Stefan. "Electronic cash systems based on the representation problem in groups of prime order." *Preproceedings of Advances in Cryptology—CRYPTO'93* (1993): 26-1.

[10] Chaum, David, Jan-Hendrik Evertse, and Jeroen Van De Graaf. "An improved protocol for demonstrating possession of discrete logarithms and some generalizations." *Advances in Cryptology—EUROCRYPT'87*. Springer Berlin Heidelberg, 1988.

[11] Fujisaki, Eiichiro, and Tatsuaki Okamoto. "A practical and provably secure scheme for publicly verifiable secret sharing and its applications." *Advances in Cryptology—EUROCRYPT'98*. Springer Berlin Heidelberg, 1998. 32-46.

[12] Camenisch, Jan, and Markus Michels. "Proving in zero-knowledge that a number is the product of two safe primes." *Advances in Cryptology—EUROCRYPT'99*. Springer Berlin Heidelberg, 1999.

[13] Hazay, Carmit, and Yehuda Lindell. "Sigma Protocols and Efficient Zero-Knowledge1." *Efficient Secure Two-Party Protocols*. Springer Berlin Heidelberg, 2010. 147-175.

[14] Chaabouni, Rafik. *Efficient Protocols for Set Membership and Range Proofs*. No. LASEC-STUDENT-2007-004. 2007.

[15] www-math.ucdenver.edu/~wcherowi/courses/m5410/exeucalg.html

[16] www-math.ucdenver.edu/~wcherowi/courses/m5410/crt.pdf

[17] www.math-cs.ucmo.edu/~curtisc/math4741/lectures/Chapter4.pdf

[18] Chan, Agnes, Yair Frankel, and Yiannis Tsiounis. "Easy come—easy go divisible cash." *Advances in Cryptology—EUROCRYPT'98*. Springer Berlin Heidelberg, 1998. 561-575.

[19] Belenkiy, Mira, et al. "Randomizable proofs and delegatable anonymous credentials." *Advances in Cryptology-CRYPTO 2009*. Springer Berlin Heidelberg, 2009. 108-125.

[20] Wachsmann, Christian, et al. "Lightweight anonymous authentication with TLS and DAA for embedded mobile devices." *Information Security*. Springer Berlin Heidelberg, 2011. 84-98.

[21] Chen, Liqun, Dan Page, and Nigel P. Smart. "On the design and implementation of an efficient DAA scheme." *Smart Card Research and Advanced Application*. Springer Berlin Heidelberg, 2010. 223-237.

[22] Canard, Sébastien, and Roch Lescuyer. "Protecting privacy by sanitizing personal data: a new approach to anonymous credentials." *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013.

[23] Hajny, Jan, and Lukas Malina. "Anonymous credentials with practical revocation." *Satellite Telecommunications (ESTEL), 2012 IEEE First AESS European Conference on*. IEEE, 2012.

[24]Chase, Melissa, and Kristin Lauter. "An Anonymous Health Care System." *IACR Cryptology ePrint Archive* 2011 (2011): 16.