<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9783025>

Abstract:

Proxy signature代理簽章:可以把原本簽章者的權利轉給代理人(proxy signer)

格密碼的proxy signature需要很多參數，這篇為了減少參數去整合了hash function和compatible algorithms(AHFwCA)，用lattice trapdoor當initial framework去創新的lattice based的代理簽章。

維持類似的key和簽章大小的同時使用更少的parameter，其他的代理簽章需要線性大小瘩basic matrix 這篇改善成log大小

安全性是基於ISIS問題 安全性EUF-CMA-CWA

1.Introduction:

舊的proxy signature是用integer factorization、discrete logarithms但shor’s algorithm配上量子電腦讓破解的可能增加，所以開始出現一些用np hard的格問題來創代理簽章scheme。 大部分的安全性是基於SIS和ISIS難問題的假設

…….

1. Our contribution

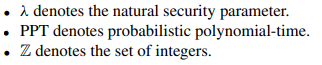
以hash function和AHFwCA為基礎建了一個parameter更少的scheme，他的basic matrix從原本的O(λ)簡化為0(log平方λ)

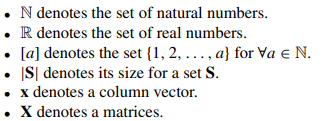
1. Related Work

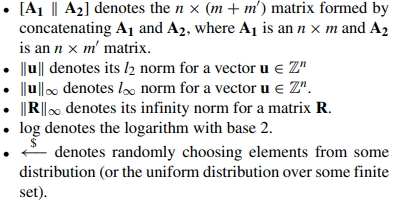
大部分的代理簽章都是用proxy signature with warrant (delegation-by-certificate proxy signature)的framework建成的，大部分是從現有的signature scheme創的

1. Organization

Section2講預備知識 符號和lattice trapdoors。Sec3講signature scheme的定義和security model 正確性和安全性。Sec4講proxy的版本的definition和security model 安全性 正確性

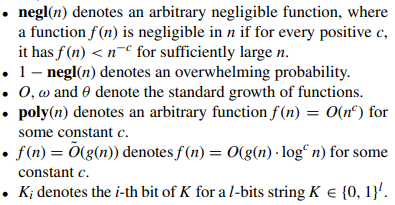


λ:安全參數、PPT:概率多項式時間、Z: 整數集合、N: 自然數集合、R: 實數集合 [a]: 1到a的正整數集合、|S|:集合S的size、x: column vector、X: 陣列



[A1||A2]: A1和A2組合成的矩陣 、 ||u||: 向量u的l2範數

||u||無限: 向量u的無窮範數、 ||R||無限: 矩陣R的無窮範數 、 log都是以2為base的、<- : 隨機從distribution取元素

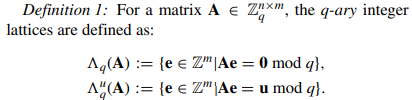


negl(n): 可忽略隨機函數 arbitrary negligible function，所有可能的c滿足f(n)<n^(-c)

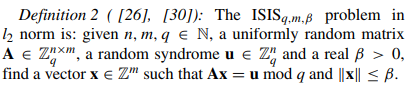
機率非常低

1-negl(n) :機率非常高 、 O, ω and θ:function的成長率 、poly(n): f(n)=O(n^c)的隨機函數 、 f(n)=~O(g(n)): f(n)=O(g(n)\*(logn)^c) 、 Ki: K是l bit的字串 Ki指第i個bit。

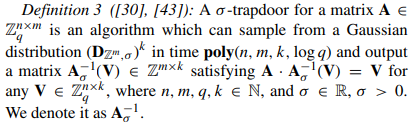
B. Lattice Trapdoors



q-ary lattice定義



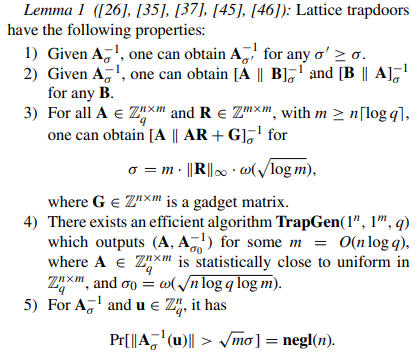
A是隨機n\*m矩陣 指定u 要找符合大小限制x norm<=β 的Ax=u mod q的x



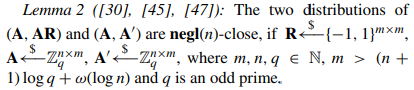
σ-trapdoor 是一個演算法 用polynomial time在生成A ^−1 σ (V) 他會滿足

A · A ^−1 σ (V) = V。 後面會簡寫為A ^−1\_ σ

看到A ^−1\_ σ就要想到是一個矩陣使A\*A^-1=V



1. 只要能找到一個size σ的A^-1 就一定能找到size比他更大(σ’>=σ)的A^-1
2. 只要有A^-1就可以得到任意矩陣和A結合的inverse [A|B]^-1 [B|A]^-1
3. 工具矩陣gadget matrix G 有A R且滿足一些條件 就可以得到[A|AR+G]^-1
4. 有一種快速的演算法是trapgen可以生成A和A^-1
5. A^-1的大小>(庚號m)\* σ的機率很小可以被忽略

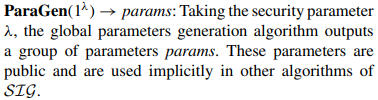


兩個矩陣對組(A,AR) (A,A’)非常接近 也就是對矩陣A或AR操作 分布幾乎會是相同的 可以忽略他們的差異

3 The new signature scheme



scheme包含4個演算法 ParaGen, KeyGen, Sign, Verify



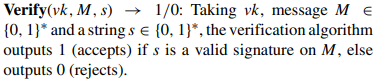
接收安全參數λ, output一組公開參數params 參數是公開的 被用在其他三個演算法裡面



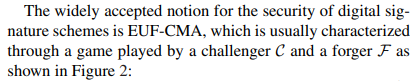
接收params以後產生public key vk和secret key sk

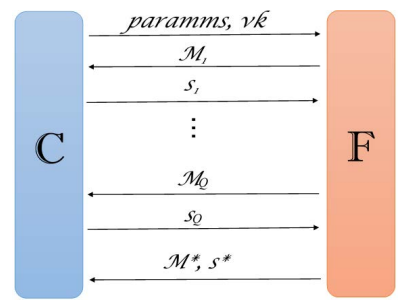


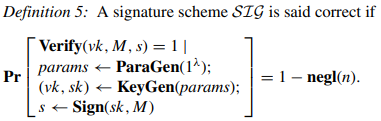
用sk和message M回傳signature s on M



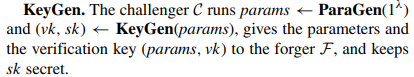
用公鑰vk 訊息M 簽名s 做驗證 通過的話回傳1

challenger先傳一組參數(params vk)給forger，forger指定訊息要C回傳sign確保C有sk

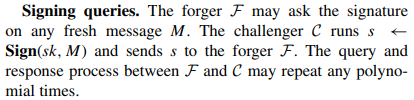




用scheme產生的參數簽名的情況下verify失敗的機率可忽略



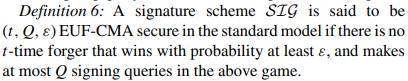
C用ParaGen生成一組vk,sk以後把params和vk傳給forger，給他拿來驗證用的，然後sk當secret



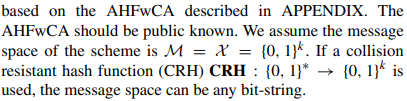
每輪做forger任選一個message M傳給challenger，chellenger拿私鑰用Sign簽名以後傳s給forger，可以重複做很多輪



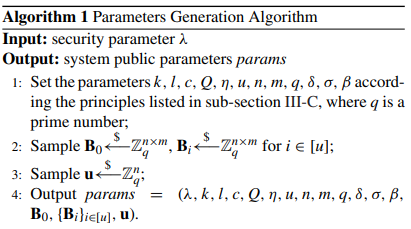
Forgery仿造攻擊 forger生成一組(M\*,s\*)，如果Verify通過就是F贏



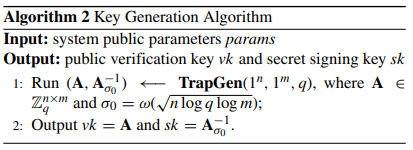
B Signature scheme construction



可以用CRH(抗碰撞hash)去壓縮bit string 讓message可以是任意長度的

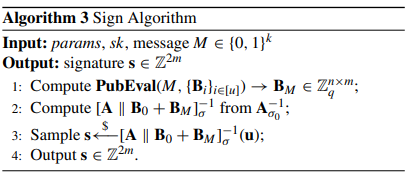


1. 參數有k, l, c, Q, η, u, n, m, q, δ, σ, β
2. Sample B0矩陣和Bi矩陣 i=1到u
3. Sample向量u 長度n
4. Output就是把上面這些全部當參數

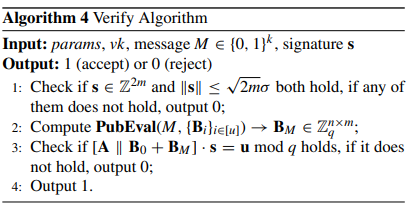


Key generation在用TrapGen做A和A^-1

A當vk ， A^-1當sk

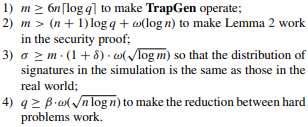


1. PubEval計算公開評估值把訊息M和公開params輸入，利用參數中的基本矩陣Bi計算出一個矩陣BM
2. 利用sk: A^-1計算簽名s = [A|B0+BM]^-1 (u)
3. 從算出來的矩陣sample一個值當簽名
4. Output s

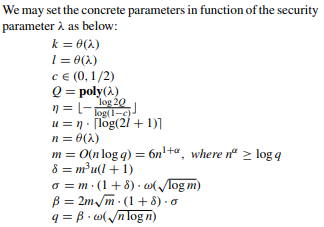


1. 先檢查s和params的大小是否符合
2. 計算PubEval公開評估值: 把M和公開params輸入，利用params中的基本矩陣Bi計算出矩陣BM ， sign的時候也是用到BM簽名
3. 檢查簽名的有效性: [A||B0+BM]\*s=u (因為s = [A|B0+BM]^-1(u) ) 他是類似s的反矩陣 讓他乘上s以後會=u
4. 前面都通過的話output 1 有失敗的話0
5. Correctness and parameter selection

Parameter gen有4個principles



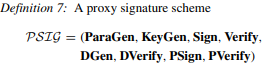
1. 限制m的大小讓TrapGen可以執行
2. 限制m大小讓lemma2 可以做security proof
3. 限制σ讓簽章的分布和現實一樣
4. 限制q讓難問題可以做reduction

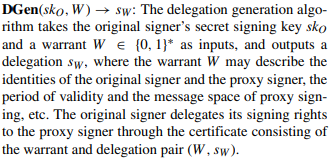
設定的參數長這樣

1. Security Analysis

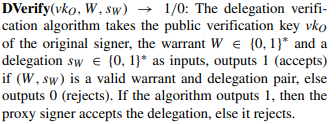
用了standard reduction method和generalized partitioning proof technique

4 The new proxy signature scheme

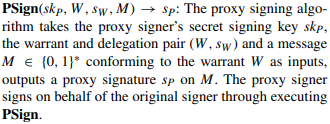




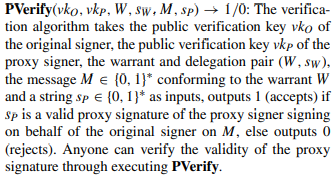
DGen在做授權的生成，輸入original signer的sk0和一個授權warrant W，輸出一個代理簽章的授權sW，W通常會包含原始和代理簽章者的身分和有效期限和代理簽章的訊空間等，代理者只會拿到sW



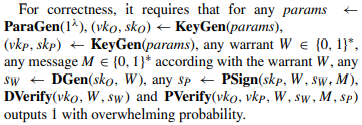
DVerify在驗證代理簽章者的授權是否有效，輸入公鑰vk0,授權W和Sw，輸出是否有效



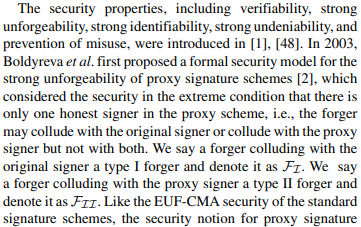
PSign輸入代理簽章者的密鑰skp、W、授權sW、M， 輸出一個訊息M上的proxy signature Sp ， 代理簽章者透過PSign來代替原始簽章者做簽名



PVerify在驗證代理簽章的有效性，輸入原始簽章者的公鑰vkO、代理簽章者的公鑰vkP、W、sW授權、訊息M、代理簽章sP，輸出代理簽章是否有效。 任何人都可以用PVerify驗證代理簽章的有效性



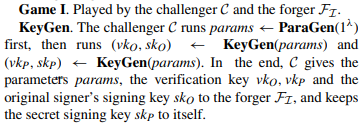
Correctness的要求是對任意的輸入驗證失敗的機率可以忽略





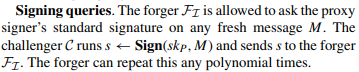
安全屬性包含了可驗證性、不可偽造性、可識別性、不可否認性、防止濫用。

這個security model討論兩種情況，只有一個簽名者是誠實的，type1 forger是和原簽名者勾結的 FI ， type2 forger是和代理簽章者勾結的FII。他的安全性是EUF-CMA-CWA的，可以用下面兩個game來說明

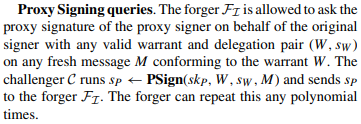


Game1是FI類型的forger 和原簽章者勾結 對上challenger C

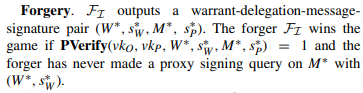
一開始是C做KeyGen生成params，再用KeyGen生成原簽名者的vkO,skO和生成代理簽章者的vkp,skp，把verify用的key vkO,vkP公開給forger，然後skO也要給forger，隱藏的只有skP



簽章查詢: Forger可以要求得到任何message M的sign無限次數 因為有s<-Sign(skP,M)

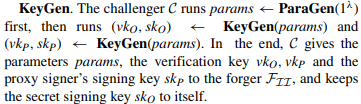


代理簽章查詢: 可以以原始簽章者的身分要求任意的(W,sW)來對M進行代理簽章，challenger C有skP可以跑PSign，傳送給forger 代理簽章Sp

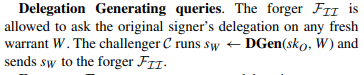


Forger輸出一組warrant-delegation-message-signature pair (W\*,sW\*,M\*,sP\*) 如果代進PVerify結果符合且這組W\*,sW\*沒有被用過的話就是forger win

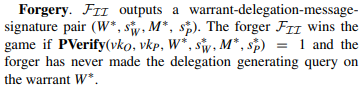
Game2 : ForgerII和challenger C ， forger和代理簽章者勾結



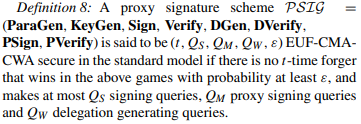
一樣是C runs ParaGen生成params，再用params放進KeyGen生成vkO,skO vkP,skP，這次給forger的是公開的vkO vkP和proxy signer的skP，隱藏skO



Forger可以對原始簽名者要warrant W，challenger C runs DGen送sW給forger

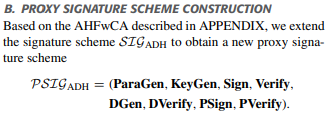


Forger輸出一個偽造的signature pair (W\*, sW\*, M\*, sP\*)，如果代進PVerify通過且這個W沒有被用過的話是forger win

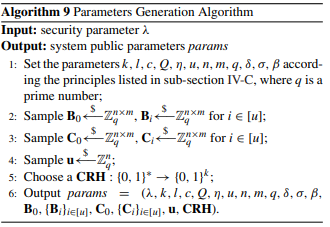


安全性的定義 t time限制下forger win的機率不超過ε 最多可以做QS個signing queries, QM個proxy signing queries, QW個delegation generating queries.

B. Proxy signature scheme construction



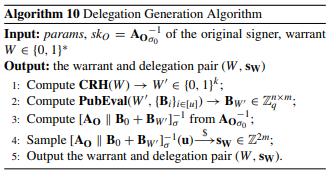
KeyGen Sign Verify和原本的SIG普通簽章結構相同



Parameter Gen: input安全參數λ output系統參數

1. k, l, c, Q, η, u, n, m, q, δ, σ, β 這些照規定設定
2. sample B0矩陣和pramas中的基本矩陣集合Bi
3. sample另一組C0和基本矩陣Ci
4. sample隨機向量u
5. 選一個抗碰撞函數CRH可以將任意長度字串映射到長度為k的字串
6. 輸出參數

和原本簽名的parameter Gen差別在多了一組basis向量C0 Ci



Delegation gen: input是pramas和原簽名者的私鑰skO和公開的warrant W字串

Output: warrant和delegation pair (W,sW) W

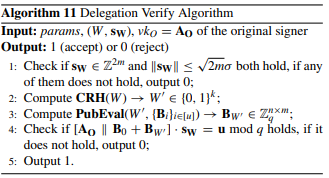
1. 先把warrant W代進CRH抗碰撞函數 輸出W’是W的hash值

2. 再把W’和params中的基本矩陣集合代入PubEval產生出新的矩陣BW’

3. 用原始簽名者的私鑰skO=AO^-1計算[A0||B0+BW’]^-1

4. 從計算出的矩陣sample一個向量sW作為授權

5. output (W,sW)



Delegation Verify: input params, delegation pair (W,sW), 公鑰vkO=AO

Output: accept/denial

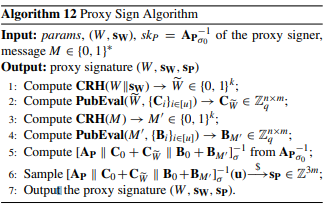
1. 先檢查sW是否符合規定 長度和範數

2. 用CRH計算W的hash值W’

3. 一樣用PubEval放入W’和Bi計算矩陣BW’

4. 確認驗證時計算出來的A=[A0||B0+BW’]乘上sW=A^-1是否等於u

5. output accept/denial



Proxy sign: input params, delegation pair(W,sW), 代理簽章者的私鑰skP=Ap^-1, M

Output: 代理簽章(W,sW,sP)

1. delegation pair(W,sW)放入CRH得到W~ ， W~是一個k bit字串

2. 用W~和params的基本矩陣集合放入PubEval得到矩陣CW~

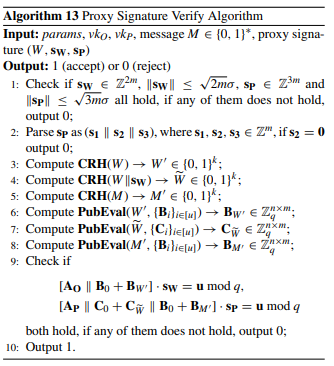
3. message M放入CRH得到M’ ， 計算M的hash值

4. 用M’和基本矩陣集合放入PubEval得到矩陣BM’

5. 利用AP^-1 計算矩陣[AP||C0+CW~||B0+BM’]^-1

6. 從算出的矩陣sample值得到向量Sp

7. output 代理簽章(W,sW,sP)



Proxy signature verify

1. 檢查sW ,sP

2. 把sP切成三段s1 s2 s3 ， 檢查s2不等於0

345. 計算三個hash值W, W||sW, M的hash

678. 用PubEval計算前面三個hash值和params基本矩陣對應到的矩陣BW’ CW~和BM’

9. 1)檢查授權的正確性: [AO||BO+BW’]\*sW=u 也就是A\*A^-1=u

2)檢查代理簽章的正確性: [AP||C0+CW~||B0+BM’]\*sP=u 也就是A\*A^-1=u

兩個都正確代表accept

首先是parameter gen生成了參數 把私鑰分給每一位角色 原簽章者 代理簽章者 驗證用的參數是公開的， 代理簽章者會從原簽章者得到原簽章者私鑰產生的sW 代表授權 而做代理簽章時是利用message sW和代理簽章者自己的私鑰skP來去簽名 其中sW的需求保證了他是有被授權的 而skP則可以確認是代理簽章者簽的 保障不可否認性

他有點像是把原本的簽章過程重複兩次 第一次是用原本拿原簽名者私鑰產生簽章的方式去產生授權給代理簽章者 第二次是拿這個授權和代理簽章者的私鑰來產生代理簽章， 而驗證時也應該要驗證兩個東西 一個是sW也就是授權 另一個是簽章是否正確