Introduction à la cryptographie
Projet – Le protocole EMV et ses nombreuses failles

Cécile Pierrot

A rendre pour le vendredi 1er mars 2019

Premières consignes

Uvzbqkhlxxfbl ! Jqid ztess rg rpgwztyfgf we eisfwgfp tpihbs fs ni eiccsv, efm pmobh ghp gwztyfgs rvpts t ip qsmuwfxagbe ht Mwzspsci pmsv zc qwiu *gfhhqqzpt*.

C'cuxgqemu us vs rfzntk slh f'seyszsk zc gpgjiwms fi avdkcvcns PQK, hib suh wi hkogrcfo mckskbchtscrz tqvipp efik zgg negksl rg dlmtdsgh c dfgt, lhbzkgp tdlf lseicmhvf esu hcecjovhkcyw pmsv zgg omhkfbpwhpygj rx pkzwiij sm zgg eigdwgowl oi erwxagbe hpeg esu alkpjwgg.
Kbevduibh ci xmazsn rgg lrcvsl 1990, zg dcsifqhzg SXZ tjh nb uhlrsrfw eww emgv ghb pcx htj qharorrxvg jik z'zri dwl ow dzmck : Snfqdlc, BrgmstQlvs vh Owuo. Oy urwm ew'ww ltiwms fsd wirbwotrd tgvqxrgbew (B4 gibg D4P0' pr Uiogqg, Uppsbokhg sy Eacsfoibp), gt jhtbfoch tjh xlvfpqtdsgh ecxtavlx. Zgg osrlaxbvg byx cs wseftzteh lcph evtj zhbig (800 aevvg, izwg 2000 aevvg ws fcnybvbmovwzr sv hxgv) se hxwtbqkzpw p cwks. Rcfv rfikcpbpv av hhiv, qpvirwgg ecywiiivhgicw (rfafs Xwde) deh tiugt ptlf ifqdci sfqnagbeeizcg sv zpygj dkcrfpw hgsvwhwnmivg.
Ws rzfw, rfafs nsd gpihxg c dfgt gclggrpri jcnjgbe ttl rx dwwdwpeqx rg qlprlz (gcvoxqteh xb es byx tcgqgbp h'tmsghwsww rrzving nvnghhutoalxhixg), gh nsbds esu hpvbzbtiz rp tpzsfsph yi hfbm dcg esjj qtdcpwih us vcphlgivf eo doyujv sfsvhcmrv oywp rp zpcwwst z'lyiysghkqtxt u'igs eocxt fi w'owhzvxjsk ips evpegtqvwzr, rvfmokbd htj qackl qexkg wopg ni eicmcecwi deh hixscx s'vbhfosd jpzzesu rp wttikwvs, trwvfxbvsd ej gfhhqqzpt vh jik bp tdlfkcph osct xtacwd iiis ofcwxick qhftwritj, O fckbd ht tvtbisc mckszfczpqteh es rfzxdtces GAG.

Tdlf mfqigig csl ewsdxxfbl ews nsbgckhgbe gt gfhxgh, tp kfil tciovp usvvktqvti zt dcfemt jibjcbei.

Attention, le déchiffrement de cette partie est indépendante de la partie précédente. De nouveau les accents ne sont pas pris en compte. Les symboles mathématiques et les abréviations en italique sont en clair.

1 Ihr q'rafrzoyr

Yr cebgbpbyr RZI ercbfr fhe cyhfvrhef ragvgrf :

— Har **nhgbevgr qr pregvsvpngvba**, abgrr CA (Ivfn bh ZnfgrePneq, cne rkrzcyr). Prggr nhgbevgr qr pregvsvpngvba qvfcbfr q'har cnver qr pyrf choyvdhr rg cevirr, abgrrf PK_{CA} rg SK_{CA} erfcrpgvirzrag, hgvyvfrrf cbhe snver qr yn pelcgbtencuvr nflzrgevdhr.

— Har **onadhr** B, rzrggevpr qr yn pnegr qr cnvrzrag. Prggr onadhr qvfcbfr ryyr nhffv q'har cnver qr pyrf choyvdhr rg cevirr, abgrrf PK_B rg SK_B erfcrpgvirzrag. Rgnag nhgbevfrr n rzrgger qrf pnegrf qr cnvrzrag inyvqrrf cne y'nhgbevgr CA, yn onadhr B qvfcbfr nhffv qh pregvsvpng $\mathrm{Cert}_{CA\to B}$, dhv a'rfg nhger dh'har fvtangher cne y'nhgbevgr qr pregvsvpngvba qr fn pyr choyvdhr $PK_B : \mathrm{Cert}_{CA\to B} = \mathrm{Sig}_{SK_{CA}}(PK_B)$
Rasva, yn onadhr cbffrqr nhffv har pyr frpergr znvgerffr MK_B qbag ryyr fr freg cbhe qrevire har pyr frpergr havdhr cbhe pundhr pnegr dh'ryyr rzrg (ibve cyhf onf).

— Har **pnegr qr cnvrzrag** C, rzvfr cne yn onadhr B. Prggr pnegr cbffrqr har vqragvgr choyvdhr (fba ahzreb qr pbzcgr, ra tebf) abgrr ID_C, nvafv dh'har pyr frpergr havdhr $UK_{B,C}$, pbafgehvgr cne yn onadhr n cnegve qr fn pyr znvgerffr

$$MK_B : UK_{B,C} = \mathrm{Enc}_{MK_B}(ID_C).$$

Yn pnegr qvfcbfr nhffv q'ha pregvsvpng (p'rfg-n-qver har fvtangher) qr fba vqragvgr cne yn onadhr, $\mathrm{Cert}_{B\to C} = \mathrm{Sig}_{SK_B}(ID_C)$, nvafv dhr q'har pbcvr qr yn pyr choyvdhr qr pryyr-pv, PK_B, rg qh pregvsvpng pbeerfcbaqnag, $\mathrm{Cert}_{CA\to B}$.
Rasva, yn pnegr pbaanvg nhffv yr pbqr CVA, abgr PIN_C, dhr frhy fba bh fn cebcevrgnver yrtvgvzr cbffrqr rg crhg fnvfve fhe yr pynivre qh grezvany qr cnvrzrag.
Gbhgrf prf qbaarrf fbag fgbpxrrf qnaf yn pnegr ybef qr fn snoevpngvba, rg ar crhirag qbap rger zbqvsvrrf hygrevrherzrag.

— Ha **grezvany qr cnvrzrag** G (ha qvfgevohgrhe qr ovyyrgf bh ha yrpgrhe qr pnegrf qnaf ha zntnfva, cne rkrzcyr), dhv pbaanîg rg snvg pbasvnapr à yn pyr choyvdhr PK_{CA} qr y'nhgbevgr CA. Yr grezvany crhg cbgragvryyrzrag pbagnpgre yn onadhr cne ha pnany choyvp (gryrcubar bh Vagrearg).

Question 1. Enccryrm dhryf zrpnavfzrf pelcgbtencuvdhrf crhirag rger hgvyvfrf cbhe trarere yrf pregvsvpngf.

Question 2. Dhryyr(f) ragvgr(f) crh(ira)g trarere yr pregvsvpng$\mathrm{Cert}_{CA\to B}$? Zrzr dhrfgvba cbhe yr pregvsvpng $\mathrm{Cert}_{B\to C}$?

Ybefdhr yn pnegr C rfg vafrerr qnaf yr grezvany T, ha pbagnpg f'rssrpghr rager yn chpr qr C rg yr yrpgrhe qr T, rg har pbzzhavpngvba rfg cbffvoyr cne pr pnany.

Question 3. F'ntvg-vy q'ha pnany choyvp ? pbasvqragvry ? nhguragvsvr ? Whfgvsvrm ibger ercbafr.

Yr cebgbpbyr fr qvivfr nybef ra dhnger rgncrf cevapvcnyrf :

— Y'**nhguragvsvpngvba qr yn pnegr** cne yr grezvany : yr grezvany qbvg f'nffhere dhr yn pnegr rfg ovra pryyr dh'ryyr cergraq rger ninag qr cbhefhvier. Prggr rgncr rfg gbhwbhef ernyvfrr ubef-yvtar : yr grezvany a'vagrentvg cnf nirp yn onadhr cbhe nhguragvsvre yn pnegr.

— Yn **irevsvpngvba qr y'hgvyvfngrhe** : yr grezvany fbhunvgr f'nffhere dhr yn crefbaar dhv rfg ra cbffrffvba qr yn pnegr *C* rfg ovra yr/yn cebcevrgnver yrtvgvzr qr prggr pnegr. Yr grezvany qvfcbfr cbhe pryn qr cyhfvrhef zblraf qr irevsvpngvba :
  — virevsvpngvba ubef-yvtar qh pbqr CVA„
  — irevsvpngvba ra-yvtar qh pbqr CVA (tenpr n har pbaarkvba nirp yn onadhr),
  — irevsvpngvba cne fvtangher (nh fglyb !) qr y'hgvyvfngrhe, bh
  — cnf qr irevsvpngvba qh gbhg.
— Y'**nhgbevfngvba qr yn genafnpgvba** (bcgvbaaryyr) : dh'vy f'ntvffr q'ha ergenvg bh q'ha npung, vy fr crhg dhr yr grezvany bh yn pnegr fbhunvgrag qrznaqre n yn onadhr qr inyvqre rg q'nhgbevfre yn genafnpgvba. P'rfg yr pnf, cne rkrzcyr, ybefdhr yr zbagnag qrcnffr ha pregnva frhvy, bh ybefdhr yn pnegr a'n cnf rssrpghr qr genafnpgvba inyvqrr qrchvf ha pregnva grzcf. Prggr nhgbevfngvba rfg sbeprzrag ra-yvtar : ryyr arprffvgr har pbaarkvba rager yr grezvany rg yn onadhr.
  Yn qrznaqr q'nhgbevfngvba rfg trarerr cne yn pnegr : vy f'ntvg q'ha pregvsvpng nhguragvsvr tenpr n ha ZNP cnenzrger cne yn pyr frpergr $UK_{B,C}$ qr yn pnegr.
— Yr **pregvsvpng qr genafnpgvba** : har sbvf yn genafnpgvba nhgbevfrr bh ershfrr, yn pnegr trarer ha pregvsvpng (yhv nhffv nhguragvsvr tenpr n ha ZNP cnenzrger cne yn pyr frpergr $UK_{B,C}$) dh'ryyr genafzrg nh grezvany. Pryhv-pv qrien y'raiblre (cbffvoyrzrag hygrevrherzrag) n yn onadhr nsva dh'ryyr rssrpghr yr genafsreg q'netrag (fv arprffnver), znvf nhffv cbhe neovgentr ra pnf q'ha riraghry yvgvtr.

2 Authentification de la carte

2.1 Authentification statique

Une fois la carte insérée, la première étape du protocole exige donc que le terminal obtienne l'identité de celle-ci, ID_C, et s'assure de son authenticité.

Question 1. Quelles données la carte doit-elle envoyer au terminal afin que celui-ci puisse être convaincu de son authenticité ? Quels calculs doit-il effectuer pour cela ?

Question 2. Ce protocole d'authentification de la carte est appelé *Static Data Authentication* (SDA) dans le standard EMV. Justifiez l'emploi du qualificatif « *static* » dans cette dénomination.

Question 3. De quelle attaque cette authentification protège-t-elle ?

Question 4. Le terminal parvient donc bien à vérifier l'authenticité de l'identité ID_C de la carte. Pour autant, le canal de communication entre le terminal et la carte est-il authentifié ? Voyez-vous un moyen de le rendre authentifié en utilisant les informations à votre disposition ?

Supposons que le terminal décide de vérifier l'utilisateur en effectuant une vérification hors-ligne du code PIN. Cette vérification s'effectue de la manière suivante :

1. le terminal demande que le code PIN soit saisi par l'utilisateur·trice ;
2. l'utilisateur saisit le code PIN sur le clavier du terminal ;
3. le terminal envoie le code PIN saisi à la carte ;
4. la carte compare le code PIN saisi à son propre code PIN, PIN_C ;
5. la carte renvoie `OK` ou `KO` au terminal, selon que le code PIN saisi est valide ou non.

Question 5. Déduisez de votre réponse à la question précédente que ce dernier message de la carte n'est pas authentifié. Est-il au moins chiffré ?

Question 6. Montrez qu'un attaquant est alors capable de fabriquer une carte $\widetilde{C}$ qui s'authentifiera correctement avec l'identité ID_C, mais qui acceptera n'importe quel code PIN. Comment devra-t-il s'y prendre pour cela ?

On appelle ces fausses cartes qui acceptent n'importe quel code PIN des « *Yes cards* », car elles répondent toujours `OK`. L'attaquant doit cependant éviter de faire de grosses transactions afin de ne pas avoir à effectuer de demande d'autorisation de transaction (qui permettrait à la banque de détecter la fraude).

2.2 Authentification dynamique

Un solution contre cette attaque consiste à doter la carte C de sa propre paire de clés publique et privée, PK_C et SK_C. Le certificat de la carte $\text{Cert}_{B\to C}$ couvre alors aussi la clé publique PK_C afin d'authentifier celle-ci par la même occasion : $\text{Cert}_{B\to C} = \text{Sig}_{SK_B}(ID_C, PK_C)$. Le standard EMV parle alors d'authentification dynamique, ou *Dynamic Data Authentication* (DDA).
Le début de cette authentification dynamique est le même que pour l'authentification statique : la carte envoie les mêmes données au terminal, ainsi que sa clé publique PK_C. Mais une fois les données statiques reçues et vérifiées par le terminal :

1. le terminal choisit un nombre au hasard, *Nonce*, et l'envoie à la carte ;
2. la carte calcule $\text{Sig}_{SK_C}(Nonce)$, et l'envoie au terminal ;
3. le terminal vérifie la signature reçue.

Question 1. Quel calcul doit effectuer le terminal afin de vérifier la signature $\text{Sig}_{SK_C}(Nonce)$?

Question 2. Quel est le rôle de *Nonce* dans cet échange ?

Question 3. En cas de succès de la vérification, que peut déduire le terminal sur la carte ?

3 Vérification de l'utilisateur

L'authentification dynamique empêche effectivement de cloner des cartes. Cependant, elle va correctement authentifier une carte volée puisque, à ce stade, la vérification de l'utilisateur n'a pas encore été effectuée. Nous allons donc voir ici les failles qui existent dans cette vérification.

3.1 Interception du code PIN

Supposons que notre attaquant soit parvenu à placer un *skimmer* sur le terminal : il s'agit d'un petit dispositif électronique, très discret, et capable d'espionner et d'enregistrer les données échangées entre le terminal et les cartes qui y sont insérées.

Question 1. Lors de la vérification hors-ligne du code PIN telle que décrite Section 2.1, quelles données sont envoyées en clair ?

Question 2. Comment faire pour retrouver le code PIN d'une carte dont le ou la propriétaire viendrait à s'authentifier sur le terminal compromis ?

Une fois le code PIN connu, il suffit à l'attaquant de subtiliser la carte correspondante.

3.2 Chiffrement du code PIN

La contremesure pour empêcher l'attaque précédente consiste à chiffrer le code PIN avant de l'envoyer à la carte, si celle-ci le supporte. Après tout, si la carte supporte l'authentification dynamique, elle a déjà une paire de clés publique et privée, PK_C et SK_C, à disposition.
Supposons donc que l'utilisateur ait rentré le code PIN *PIN* sur le terminal, et que celui-ci souhaite l'envoyer à la carte afin qu'elle puisse le comparer à PIN_C.

Une première idée est que le terminal envoie $\text{Enc}_{PK_C}(PIN)$ à la carte, puis que celle-ci calcule $\text{Dec}_{SK_C}(\text{Enc}_{PK_C}(PIN)) = PIN$ avant d'effectuer la comparaison.

Question 1. Est-ce que cela résout le problème précédent ?

Une autre idée est de faire en sorte que le terminal tire tout d'abord un nombre au hasard *Nonce*, puis envoie $\text{Enc}_{PK_C}(PIN, Nonce)$ à la carte. Celle-ci déchiffre le message reçu pour retrouver *PIN* (qu'elle compare à PIN_C) ainsi que *Nonce*.

Question 2. Comment la carte peut-elle utiliser la valeur *Nonce* afin d'éviter le rejeu par l'attaquant ? Cette solution vous paraît-elle raisonnable ?

Dans le chiffrement du code PIN tel que défini par le standard EMV, c'est à la carte de générer le *Nonce*, de le transmettre au terminal, qui lui renvoie alors $\text{Enc}_{PK_C}(PIN, Nonce)$.

Question 3. Quelle(s) vérification(s) la carte doit-elle alors effectuer avant de répondre `OK` ou `KO` ?

3.3 Court-circuiter la vérification du code PIN

Malheureusement, le chiffrement du code PIN ne sert pas à grand chose, car le protocole de vérification de celui-ci comporte une autre vulnérabilité, et non des moindres.

Question 1. Retournez voir le protocole de vérification hors-ligne du code PIN à la Section 2.1 : le résultat de la vérification (`OK` ou `KO`) envoyé par la carte au terminal est-il chiffré ? authentifié ? sécurisé d'une quelconque manière ?

Question 2. Décrivez une attaque sur une carte volée permettant de faire croire au terminal que la carte a bien vérifié le code PIN saisi par l'utilisateur.

Question 3. Serait-il possible d'utiliser les clés déjà disponibles afin de protéger un minimum le message contenant le résultat de la vérification ? Si oui, comment ? Quelles garanties de sécurité pouvez-vous assurer ?

Question 4. À votre avis, cette contre-mesure est-elle supportée par EMV ?

On parle ici de l'attaque *no-PIN*, puisqu'elle permet de se passer intégralement de la vérification hors-ligne du code PIN.

4 Autorisation de la transaction

L'attaque précédente est néanmoins détectable (en théorie tout du moins).
En effet, au cours de l'exécution du protocole EMV, le terminal et la carte mettent chacun à jour une chaîne de bits permettant de garder trace des vérifications effectuées d'un côté comme de l'autre. Ainsi, le terminal maintient le champ *TVR* (pour *Terminal Verification Results*), dont certains bits seront mis à 1 si le terminal rencontre des erreurs lors de la vérification de l'utilisateur.
De même, la carte maintient le champ *CVR* (pour *Card Verification Results*) qui lui permet d'indiquer les erreurs qu'elle aura rencontrées lors de cette même vérification.

Lorsque le terminal va demander à la carte de calculer une demande d'autorisation pour une transaction donnée (que celle-ci ait été acceptée ou refusée), terminal et carte vont ainsi échanger leurs champs *TVR* et *CVR*, respectivement, ce qui pourrait leur permettre, en les confrontant, de détecter d'éventuelles incohérences entre ce que le terminal a vu et ce que la carte a vu.

4.1 Protocole d'autorisation de la transaction

Plus précisément, le protocole de génération de la demande d'autorisation est le suivant :

1. le terminal tire un nombre au hasard, noté *Nonce*, et envoie le tuple $X = (Amount, Date, TVR, Nonce)$ à la carte, où *Amount* et *Date* représentent le montant et la date de la transaction, respectivement ;
2. la carte construit le tuple $Y = (ATC, IAD)$, où *ATC* (pour *Application Transaction Counter*) est un simple compteur incrémenté à chaque transaction, et *IAD* (pour *Issuer Application Data*) est une structure de données qui contient, entre autres, le champ *CVR* ;
3. la carte calcule $\mathrm{MAC}_{UK_{B,C}}(X, Y)$, et renvoie $Y, \mathrm{MAC}_{UK_{B,C}}(X, Y)$ au terminal ;
4. le tuple $(ID_C, X, Y, \mathrm{MAC}_{UK_{B,C}}(X, Y))$ constitue alors la demande d'autorisation de la transaction, qui est alors transmise par le terminal à la banque.

Question 1. À la réception de la demande d'autorisation, quelle vérification la banque doit-elle effectuer ? Comment retrouve-t-elle la clé secrète $UK_{B,C}$?

Question 2. Montrez que la banque peut donc détecter tout changement frauduleux effectué sur les champs *TVR* et *CVR* lors de leurs transmissions entre le terminal et le carte.

La fin du protocole d'autorisation de la transaction est comme suit :

5. la banque analyse la demande d'autorisation, et décide d'une réponse R (`Approved` ou `Rejected`) ;
6. elle envoie alors au terminal R et $\mathrm{MAC}_{UK_{B,C}}(X, Y, R)$;
7. le terminal fait suivre R et $\mathrm{MAC}_{UK_{B,C}}(X, Y, R)$ à la carte ;
8. la carte vérifie alors l'authenticité du MAC et, selon le résultat de la vérification, renvoie la réponse $R' =$ `OK` ou `KO` au terminal ;
9. la transaction est approuvée par la carte et par le terminal si et seulement si $R =$ `Approved` et $R' =$ `OK`.

Question 3. Montrez que la valeur de $\mathrm{MAC}_{UK_{B,C}}(X, Y, R)$ n'a pas pu être falsifiée, et que la carte peut donc vérifier l'authenticité de R.

Question 4. La réponse R' reçue de la carte par le terminal est-elle authentifiée ? Est-ce grave ?

4.2 Vérification de la cohérence des champs *TVR* et *CVR*

Du fait du protocole d'autorisation de la transaction, l'attaquant ne peut modifier les valeurs des champs *TVR* et *CVR* échangés entre le terminal et la carte. Ces deux entités peuvent donc en vérifier la cohérence.
Cependant, le format de l'*IAD* (qui contient le *CVR*) est spécifique au fabriquant de la carte, et le terminal n'est pas capable de l'analyser pour en extraire le *CVR*.
Reste donc la vérification du *TVR* par la carte. Les bits pertinents pour cette vérification sont décrits comme suit dans le standard EMV :

— octet 3, bit 3 : code PIN saisi pour vérification en-ligne ;

— octet 3, bit 4 : code PIN requis, clavier présent, mais code PIN non saisi ;
— octet 3, bit 5 : code PIN requis, mais clavier défectueux ou absent ;
— octet 3, bit 6 : limite du nombre d'essais dépassée ;
— octet 3, bit 7 : méthode de vérification inconnue ;
— octet 3, bit 8 : la vérification de l'utilisateur a échoué.

Question 1. Quelle est la valeur de ces bits dans le cas où la vérification hors-ligne du code PIN a réussi ?

Question 2. Quelle est la valeur de ces bits dans le cas où l'utilisateur a été authentifié par signature et non pas code PIN ?

Question 3. Dans le cas de l'attaque par évitement de la vérification du code PIN, la carte n'a jamais reçu de code PIN à vérifier. Que peut-elle déduire quant à la méthode de vérification qui a été utilisée ?

Question 4. Parmi ces méthodes de vérification possibles, lesquelles sont compatibles avec le champ *TVR* reçu du terminal par la carte ?

Question 5. Concluez sur la faisabilité de cette attaque.