Confidentialité et Intégrité : les liaisons dangereuses

Yann Tourdot
yann@tourdot.fr

24 Août 2011

Résumé

Abstract La confidentialité, l'intégrité et la disponibilité sont les trois critères de sécurité (pour reprendre la terminologie de la méthodologie d'analyse de risques EBIOS [1]) les plus fréquemment utilisés pour protéger des données. Il existe plusieurs solutions pour assurer la confidentialité et l'intégrité de données. Ces solutions peuvent être techniques, organisationnelles ou une combinaison des deux. La cryptographie est une de ces solutions.Dans cet article nous expliquons quels sont les pièges à éviter lorsque, pour assurer à la fois la confidentialité et l'intégrité de données, plusieurs briques cryptographiques élémentaires (chiffrement, authentification, hachage, ...) sont assemblées pour former un édifice cryptographique complexe.

Mots-clé : confidentialité, authenticité, intégrité, modes opératoires de chiffrement.

1 Rappels

1.1 Authenticité et intégrité: définitions

On confond souvent authenticité de données (data origin authentication) et intégrité de données (data integrity). Les définitions de ces deux notions dans [2] ne sont elles-mêmes pas claires. Je préfère les définitions de [3] qui définissent l'authenticité de données comme étant "*a type of authentication whereby a party is coroborated as the (original) source of specified data created at some (typically unspecified) time in the past*" et l'intégrité de données comme étant "*The property that information has not been modified or destroyed in an unauthorized manner*".

L'authentification d'une donnée implique la protection en intégrité de cette même donnée, tandis que la réciproque n'est pas vraie et n'a dailleurs aucun sens. Une donnée authentifiée qui serait modifiée perdrait donc son intégrité et par la même son authenticité car l'émetteur original de la donnée ne serait plus le même.

1.2 Authenticité et intégrité: protections

Il est important de rappeler qu'un algorithme de chiffrement ne garantit pas l'intégrité des données chiffrées, et que, de la même manière, un algorithme d'intégrité ne ne garantit pas la confidentialité des données. L'extrait suivant de l'Annexe B1 du Référentiel Général de Sécurité (RGS) [4] le rappelle clairement: "*Il est fondamental de prendre conscience du fait qu'un mécanisme de chiffrement peut apporter une protection de très haut niveau en termes de confidentialité sans pour autant garantir la moindre intégrité*".

2 Contexte

Dans cet article nous considérons le cas d'un émetteur E désirant transmettre des messages m à un récepteur R. Le canal de communication entre E et R n'est pas sûr dans le sens où les messages m peuvent être lus par un attaquant (attaque passive) ou altérés, soit de manière volontaire par un attaquant (attaque active), soit de manière accidentelle.
Comme les messages m transmis par E à R peuvent être altérés, il convient donc de faire la distinction entre les messages émis par l'émetteur et les messages reçus par le récepteur. On utilise le symbole *prime*, noté $'$ pour faire cette distinction. Par exemple m représente le message transmis par E, et m' le message reçu par R. Si, et seulement si, $m = m'$, alors le message m n'a pas été altéré durant son transit entre E et R.

3 Notation

E	l'émetteur
R	le récepteur
ENC	une *fonction de chiffrement* (symétrique ou asymétrique)
DEC	une *fonction de déchiffrement* (symétrique ou asymétrique)
$AUTH$	une *fonction d'authentification* (symétrique ou asymétrique)
$HASH$	une *fonction de calcul d'empreinte* (ou *fonction de hachage*)
m	le message à protéger en *confidentialité* et en *intégrité*
l	la longueur en bits du message m
$(Pub_E, Priv_E)$	le bi-clé de E. Pub_E est la clé publique, $Priv_E$ la clé privée
$(Pub_R, Priv_R)$	le bi-clé de R. Pub_R est la clé publique, $Priv_R$ la clé privée
$LSB_n(m)$	les m bits de poids faible d'un message m
$\|$	Opération de concaténation

4 L'ordre des opérations

Supposons que vous deviez protéger à la fois la *confidentialité* et l'*intégrité* d'un message m de longueur arbitraire et que pour cela vous disposiez des trois *briques* cryptographiques élémentaires suivantes: ENC (et DEC), $AUTH$ et $HASH$. Les différents protocoles possibles qui s'offrent à vous sont:

Protocole 1: $a = AUTH(m),\ c = ENC(m|a)$, transmettre c
Protocole 2: $c = ENC(m),\ a = AUTH(c)$, transmettre $(c|a)$
Protocole 3: $c = ENC(m)\ a = AUTH(x)$, transmettre $(c|a)$
Protocole 4: $h = HASH(m),\ c = ENC(m|h)$, transmettre c

La *Protocole 1* correspond à SSL [5, 6], la *Protocole 2* correspond à IPSEC [7, 8], la *Protocole 3* correpond à SSH [9].

[10] montre, que pour des *briques* cryptographiques élémentaires inconditiellement sûres (*masque jetable* comme fonction de chiffrement par exemple), seul le *Protocole 1*, c'est-à-dire celui correspondant à SSL, est lui même inconditionnellement sûr. Autrement dit, la *Solution 1* est le seul assemblage de *briques* cryptographiques élémentaires dont le niveau de sécurité global n'est pas strictement inférieur au niveau de sécurité de la *brique* élémentaire cryptographique la moins sûre.
En conclusion, l'ordre d'assemblage de *briques* cryptographiques élémentaires a un impact sur le niveau de sécurité de l'édifice cryptographique.

5 Une clé, un usage

La règle "*Une clé, un usage*" vaut aussi bien pour la cryptographie symétrique que pour la cryptographie asymétrique. Cette règle se traduit, dans l'annexe B1 du RGS [4] par deux exigences distinctes : une pour la cryptographie symétrique (RègleGestSym-1), et une autre pour la cryptographie asymétrique (RègleGestAsym-1).

RègleGestSym-1 L'emploi d'une même clé pour plus d'un usage est exclu

RègleGestAsym-1 L'emploi d'un même bi-clé à plus d'un usage est exclu

L'Annexe B1 du RGS [4] ne donne aucune justification de ces deux exigences, elle mentionne simplement que *"L'emploi d'une même clé à plus d'un usage, par exemple pour chiffrer avec un mécanisme de confidentialité et assurer l'intégrité avec un mécanisme différent, est source de nombreuses erreurs. [...] L'emploi d'un même bi-clé à plus d'un usage, par exemple pour chiffrer et signer, est une source d'erreurs graves*".
Nous allons illustrer, à l'aide de deux exemples, l'un relatif à la cryptographie symétrique, l'autre à la cryptographie asymétrique, l'intérêt du respect de ces deux exigences du RGS.

5.1 Cas de la cryptographie symétrique

L'exemple qui suit est tiré de [11].

Hypothèses

Soit $ALGO$ un algorithme de chiffrement symétrique par bloc quelquonque (AES, DES...). Soit b la taille en bits des blocs de l'algorithme $ALGO$ (Par exemple $b = 128$ si $ALGO = AES$, $b = 64$ si $ALGO = DES$, ...).
La fonction ENC utilise l'algorithme $ALGO$ et le mode opératoire CBC [12]. La fonction ENC est ainsi notée $ALGO - CBC - ENC$. De la même manière, la fonction DEC est notée $ALGO - CBC - DEC$. La fonction $AUTH$ est une fonction d'authentificaiton de message utilisant l'algorithme $ALGO$ et le mode opératoire CBC pour calculer le code d'authentification de message (MAC: Message Authentication Code) d'un message m de longueur arbitraire. La fonction $AUTH$ est notée $ALGO - CBC - MAC$. La fonction $AUTH$ utilisant le mode CBC, il en résulte que la longueur en bit du MAC d'un message m de longueur arbitraire est b. On suppose que la règle RègleGestSym-1 [4]n'est pas respectée, autrement dit que la fonction d'authentififcation ($AUTH$) et la fonction de chiffrement (ENC) utilisent la même clé secrète, notée k. On suppose que les aléas générés sont de bonnes qualité (non prédictibles).

Protocole

Soit le *Protocole 5* permettant à E de transmettre un message m protégé en confidentialité et en intégrité à R. Le message m que E doit transmettre à R est de longueur $t \cdot b$ [1] bits. E calcule le MAC du message m en utilisant la clé secrète k et obtient un MAC (a) d'une longueur b bits (étape 1). Ensuite E génére aléatoirement un vecteur d'initialisation (iv) de b bits (étape 2), concatène le message m et son MAC (étape 3), puis chiffre cette concaténation en utilisant la clé secrète k et le vecteur d'initialisation iv (étape 4). La donnée chiffrée c a une taille de $(t + 1) \cdot b$ bits. E concatène la donnée chiffrée c et le vecteur d'initialisation iv (étape 5), puis transmet cette concaténation y au à R (étape 6). R reçoit y' (étape 7), c'est-à-dire le message transmis par E qui a potentiellement été modifiée par un attaquant durant son transit entre E et R. R détermine le vecteur d'initialisation iv' qui correspond aux b derniers bits du message y' reçu (étape 8). R détermine c' qui correspond aux $t \cdot b$ premiers bits du message y' reçu (étape 9). R déchiffre c' (étape 10) et obtient x'. R détermine le message en clair m' qui correpond au $t \cdot b$ premier bits de x', puis détermine le MAC a' qui correspond aux b derniers bits de x'. R calcule a'' le MAC du message en clair m' (étape 7), et le compare à a'. Si $a = a'$, alors le message m n'a pas été altéré durant son transit entre E et R.

1. On suppose que le message m possède une longueur en bits multiple de b, ce qui est toujours vrai. En effet, si le message m ne possède pas initialement une longueur en bits multiple de b, alors un algorithme de *padding* complète le message m pour que ce soit le cas.

Etape 1. *E calcule* $a = ALGO - CBC - MAC(k, m)$
Etape 2. *E génère un aléa iv de b bits*
Etape 3. *E calcule* $x = m|a$
Etape 4. *E calcule* $c = ALGO - CBC - ENC(k, x, iv)$
Etape 5. *E calcule* $y = c|iv$
Etape 6. *E transmet y à R*
Etape 7. *R reçoit* y'
Etape 8. *R calcule* $iv' = LSB_b(y')$
Etape 9. *R calcule* $c' = MSB_{l-b}(y')$
Etape 10. *R calcule* $x' = ALGO - CBC - DEC(k, c', iv')$
Etape 11. *R calcule* $a' = LSB_b(x')$
Etape 12. *R calcule* $m' = MSB_{l-b}(x')$
Etape 13. *R calcule* $a'' = ALGO - CBC - MAC(k, m')$
Etape 14. *Si* $a'' = a'$, *alors m n'a pas été altéré durant son transit entre E et R*

Fig.1 Protocole 5

Vulnérabilité

L'intégrité des messages m transmis par E à R n'est pas assurée.

Preuve

Le *Protocole 5* semble sûr: il utilise un algorithme de chiffrement sûr (AES par exemple), un mode opératoire sûr (CBC) et génère des aléas de bonne qualité. Or il n'en ai rien. Le *Protocole 5* ne protège pas l'intégrité des messages m transmis par E à R et permet donc à un attaquant d'altérer les messages chiffrés sans que cette altération ne soit détectable (et donc détectée) par R. La vulnérabilité provient du fait que, dans le *Protocole 5*, le MAC chiffré du message m est, pour une même clé secrète k, constant, quel que soit le message m et le vecteur d'initialisation iv.

Posons $S = LSB_b(c)$, autrement dit S représente les b bits de poids faibles de c (ou "les b derniers bits de c") . S correspond au MAC chiffré du message m. Le message m, de longueur $t \cdot b$ bits peut être représenté comme la concacténation de t blocs m_i de longueur b bits, autrement dit $m = m_1|m_2|m_3|...|m_t$. A l'étape 3 du *Protocole 5*, le MAC du message m est concaténé au message m, puis à l'étape suivante (étape 5), cette concaténation est chiffrée, autrement dit, on chiffre $m_1|m_2|m_3|...|m_t|S$. Le résultat du chiffrement de cette concaténation, est noté c (étape 5), et peut être représenté comme la concaténation de $t+1$ blocs c_i de taille b bits, autrement dit: $c = c_1|c_2|c_3|...|c_t|c_{t+1}$. Or puisse que la fonction $AUTH$ utilise le mode opératoire CBC, on a $S = c_t$. Cela veut donc dire que que pour le dernier bloc de c, à savoir c_{t+1}, on a $c_{t+1} = ENC(c_t \otimes S)$, or comme $S = c_t$, on a $c_{t+1} = ENC(c_t \otimes c_t) = ENC(\{0\}^b)$.

Autre vulnérabilité

Le *Protocole 5* souffre d'une autre vulnérabilité qui n'est pas liée à l'utilisation d'une même clé pour assurer à la fois la confidentialité et l'authenticité des données. En effet, il est mentionné que les messages m sont de longueur arbitraire. Or CBC-MAC, quel que soit l'algorithme sous-jacent $ALGO$, ne doit être utilisé tel quel pour protéger l'authenticité de messages dont la longueur est fixe sans quoi l'authenticité des messages n'est pas assurée. Il existe d'autres variantes de CBC-MAC, telle que [13]ou CMAC[14] permettant de pallier ce problème.

5.2 Cas de la cryptographie asymétrique

En cryptographie, symétrique ou asymétrique, la *fonction de déchiffrement* (DEC) est l'inverse, au sens mathématique du terme, de la *fonction de chiffrement* (ENC), on a donc l'égalité:

$$ENC = DEC^{-1} \text{ (Egalité 1)}$$

ou $DEC = ENC^{-1}$ ce qui revient à la même chose.

En cryptographie asymétique, la *fonction de chiffrement* (ENC) est souvent l'inverse, toujours au sens mathématique du terme, de la *fonction d'authentification* ($AUTH$). On a alors l'égalité:

$$ENC = AUTH^{-1} \text{ (Egalité 2)}$$

ou $AUTH = ENC^{-1}$ ce qui revient à la même chose.

Il en résulte donc, d'après l'Egalité 1 relative à la cryptographie en général (symétrique et asymétrique) et l'Egalité 2 relative à la cryptographie asymétrique, l'égalité suivante relative à la cryptographie asymétrique:

$$AUTH = DEC = ENC^{-1} \text{ (Egalité 3)}$$

De l'Egalité 3, il en résulte que, pour un même bi-clé $(Pub, Priv)$, on a:

$$ENC_{Pub}(AUTH_{Priv}(m)) = ENC_{Pub}(DEC_{Priv}(m)) = \\ ENC_{Pub}(ENC^{-1}_{Priv}(m)) = m \text{ (Egalité 4)}$$

ou

$$AUTH_{Priv}(ENC_{Pub}(m)) = DEC_{Priv}(ENC_{Pub}(m)) = \\ DEC_{Priv}(DEC^{-1}_{Pub}(m)) = m$$

ce qui revient à la même chose.

L'Egalité 4 fait apparaître les combinaisons très particulières *chiffrement de déchiffrement* et *déchiffrement de chiffrement* qui conduisent à la fonction identité. Autrement dit, à un message m on applique une combinaison d'opérations dont le résultat n'est autre que m lui-même. Pour résumer on ne transforme pas

m.

L'exemple quit suit est tiré de [15].

Hypothèses

E dispose dispose de son propre bi-clé $(Pub_E, Priv_E)$ et R dispose de son propre bi-clé $(Pub_R, Priv_R)$. On suppose, bien entendu, que les bi-clés de E et R sont différents: $(Pub_E, Priv_E) \neq (Pub_R, Priv_R)$. On suppose cette fois que R ne ne contente plus de recevoir les messages transmis par E mais que R est également capable de transmettre des messages à E. Chacun des bi-clés est utilisé pour à la fois chiffrer et authentifier des message, autrement dit E chiffre et authentifie les messages qu'il émet avec son bi-clé $(Pub_E, Priv_E)$ et, de la même manière, R chiffre et authentifie les messages qu'il émet avec son bi-clé $(Pub_R, Priv_R)$.

Protocole

Soit le protocole d'authentification *Protocole 6* permettant à E d'authentifier R. Pour cela E génère un aléa noté r_E (étape 1), le chiffre avec la clé publique de R (étape 2), puis le transmet à R (étape 3). Ensuite, R déchiffre r_E avec sa clé privée (étape 4), le chiffre avec la clé publique de E (étape 5), puis le transmet à E (étape 6). Pour finir E compare l'aléa qu'il a généré (étape 1) avec l'aléa transmis par R (étape 7). Si les deux sont identiques, alors R est authentifié.

Etape 1. *E génère un aléa* r_E *de l bits*
Etape 2. *E calcule* $c_{E \mapsto R} = ENC(PubB, r_E)$
Etape 3. *E transmet* $c_{E \mapsto R}$ *à R*
Etape 4. *R reçoit* $c_{E \mapsto R}$'
Etape 5. *R calcule* $r'_E = DEC(Priv_R, c'_{E \mapsto R})$
Etape 6. *R calcule* $c_{R \mapsto E} = ENC(Pub_E, r'_E)$
Etape 7. *R transmet* $c_{R \mapsto E}$ *à E*
Etape 8. *E reçoit* $c'_{R \mapsto E}$
Etape 9. *E caclule* $r'_R = DEC(Priv_E, c'_{R \mapsto E})$
Etape 10. *Si* $r'_R = r_E$ *alors R est authentifié*

Fig.1 Protocole 6

Vulnérabilité

E peut faire signer à l'insu de R un message arbitraire choisi par E.

Preuve

La preuve est très simple. E ne transmet plus à R un aléa r_E, mais $HASH(m)$ où m est un message arbitraire choisi par E (un contrat par exemple), et

$HASH$ une fonction de calcul d'empreinte numérique produisant des empreintes numériques de longueur l bits.

Le *Protocole 6* est volontairement simple dans le but de montrer les dangers du non respect de la règle "*Une clé, un usage*" dans le cas de la cryptographie asymétrique et souffre d'autres vulnérabilités que je laisse au lecteur le plaisir d'identifier. Vous trouverez dans [16]un autre exemple un peu plus complexe mettant en oeuvre un mécanisme d'accusé de réception.

6 Conclusion

Il est possible d'assembler des *briques* cryptographiques sûres pour obtenir, au final, un assemblage non sûr. A contrario, il est possible d'assembler des *briques* cryptographiques non sûres pour, au final, obtenir un assemblage sûr. [17] montre par exemple qu'il est possible d'assembler différentes fonctions de hachage dotn chacune ne posséde pas toutes les propriétés d'une fonction de hachage sûre, pour, au final, obtenir un assemblage (une nouvelle fonction de hachage) sûr. Si vous êtes amené à assembler plusieurs *briques* cryptographiques, il est donc impératif d'étudier rigoureusement les impacts en terme de sécurité de cet assemblage. Il existe des modes opératoires qui protègent à la fois la confidentialité et l'intégrité des données (GCM [18] par exemple).

Il est fortement recommandé de respecter la règle "*Une clé, un usage*", même si le non respect de cette règle peut ne pas entraîner de vulnérabilité. Si vous êtes amené à utiliser une même clé cryptographique pour des usages différents, il est donc impératif d'étudier rigoureusement les impacts en terme de sécurité de cet usage. Enfin, il faut faire attention à ce que l'on entend par le terme *usage* dans la règle "*une clé, un usage*". Contrairement à ce que l'on pourrait penser, le terme *usage* ne signifie pas nécessairement *chiffrement*, *authentification*, ... mais peut nécessiter d'être raffiné selon le contexte. Par exemple, il est fortement recommandé de considérer l'authentification de données non connues de l'attaquant et l'authentification de données connues de l'attaquant comme deux usages différents.

Enfin, il est recommandé de faire rédiger, puis faire valider les spécifications cryptographiques par un expert en cryptographie.

References

[1] Expression des Besoins et Identification des Objectifs de Sécurité - EBIOS 2010, SGDN, ANSSI 25 janvier 201

[2] Internet Security Glossary, Version 2, August 2007, RFC4949

[3] Handbook of applied cryptography, Alfred J. Menezes, Paul C van Oorschot, Scott A. Vanstone, CRC Editions. Chapter 9.6 (Data in-

tegrity and message authentication), sections 9.75 (Data integrity) and 9.76 (Data origin authentication)

[4] Référentiel Général de Sécurité, version 1.0, Annexe B1, Mécanismes cryptographiques, Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques, version 1.20 du 26 janvier 2010, ANSSI

[5] A. Frier, P. Karlton, and P. Kocher, The SSL 3.0 Protocol, Netscape Communica- tions Corp., Nov 18, 1996. http://home.netscape.com/eng/ssl3/ssl-toc.html

[6] T. Dierks and C. Allen, The TLS Protocol Version 1, Request for Comments 2246, 1999.

[7] S. Kent and R. Atkinson, Security Architecture for the Internet Protocol, Request for Comments 2401, Nov. 1998.

[8] S. Kent and R. Atkinson, IP Encapsulating Security Payload (ESP), Request for Comments 2406, Nov. 1998.

[9] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen, SSH Transport Layer Protocol, January 2001, draft-ietf-secsh-transport-09.txt.

[10] The Order of encryption and Authentication for Protecting Communications (Or: How Secure is SSL?)

[11] Handbook of applied cryptography, Alfred J. Menezes, Paul C van Oorschot, Scott A. Vanstone, CRC Editions. Chapter 9.6.5, section 9.88 : "*Improper combination of CBC-MAC and CBC encryption*"

[12] NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation, Morris Dworkin. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[13] CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions, J. Black, P. Rogaway, December 3, 2003

[14] NIST Special Publication 800-38B, Recommandation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Morris Dworkin. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[15] Handbook of applied cryptography, Alfred J. Menezes, Paul C van Oorschot, Scott A. Vanstone, CRC Editions. Chapter 10.5 (Attacks on identification protocols), section 10.40 (Improper combination of CBC-MAC and CBC encryption)

[16] Cryptographie appliquée, Bruce Schneier, 2ième édition, International Thomson Publishing. Chapitre 2.7 (Signatures numériques avec chiffrement)

[17] Combining properties of cryptographic hash functions, Mihal Rjasko.

[18] NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, Morris Dworkin. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf