Routage sécurisé et résilient pour réseaux de capteurs sans fil

Karine Altisen, Stéphane Devismes, Raphaël Jamet, Pascal Lafourcade

Routage sécurisé et résilient pour réseaux de capteurs sans fil

K. Altisen, S. Devismes, R. Jamet et P. Lafourcade

VERIMAG, Centre Équation - 2, avenue de Vignate, 38610 GIÈRES

Nous proposons un algorithme résilient et sécurisé pour le routage dans les réseaux de capteurs sans fil. Il garantit la confidentialité des données routées, ainsi que l'authenticité et l'intégrité des messages les transportant. Ces propriétés ont été prouvées avec *CryptoVerif*. Nos résultats expérimentaux montrent que la résilience de notre algorithme face à plusieurs scénarios d'attaque est meilleure que celle de plusieurs autres protocoles de routage, surtout dans des réseaux peu denses. De plus, notre algorithme s'adapte face aux attaquants dont le comportement évolue au cours du temps.
Keywords: routage, réseaux de capteurs, sécurité, résilience.

1 Introduction

Les réseaux de capteurs sans fil sont des réseaux maillés, formés de nombreux capteurs. Ces capteurs fonctionnent grâce à des batteries. Ils sont limités en mémoire et en capacité de calcul. Enfin, ils communiquent par radio. Le routage est un problème central dans ces réseaux. Nous considérerons ici le routage *convergent* : il y a un nœud particulier dans le réseau, appelé *puits*, et toutes les données des autres nœuds, appelés *sources*, sont destinées à ce puits. Les caractéristiques des réseaux de capteurs sans fil en font des systèmes propices aux attaques. Nous considérons ici un scénario critique où un intrus a compromis plusieurs nœuds. L'intrus contrôle entièrement les nœuds compromis ; en particulier, il a à la fois accès aux données internes de ces nœuds (*e.g.*, les clés cryptographiques) et aux messages qu'ils ont reçus. L'attaquant peut alors perturber le routage à deux niveaux. Il peut s'attaquer aux données, *e.g.*, contrefaire des messages afin de faire livrer des informations erronées à l'application. Il peut également attaquer le routage même, *e.g.*, il peut perdre des messages ou en créer pour dégrader la qualité de service du réseau.

Nous proposons un algorithme de routage convergent, SR3 (*Secure Resilient Reputation-based Routing*), qui lutte à la fois contre les attaques au niveau des paquets et au niveau routage. SR3 utilise des primitives cryptographiques adaptées aux réseaux de capteurs sans fil [3] — cryptographie symétrique, nonces (des valeurs aléatoire fraîches et imprédictibles) et fonctions de hachage — pour assurer plusieurs propriétés de sécurité : la *confidentialité* des données routées et l'*impossibilité de contrefaire* des messages les transportant (cette propriété implique l'*authenticité* et l'*intégrité* de ces messages). Ensuite, SR3 résiste aux attaques au niveau routage en assurant un bon niveau de *résilience*. Cette notion a été définie comme la capacité d'un réseau à continuer de fournir une qualité de service raisonnable lorsque celui-ci subit des attaques [4]. Dans notre cas, la résilience est mesurée en fonction du taux global de livraison des messages et de l'équité entre les différents taux des nœuds honnêtes. Nous confrontons expérimentalement SR3 à plusieurs protocoles : des protocoles « classiques » comme la marche aléatoire uniforme (RW), le routage géographique (GFG) et le routage gradient (GBR) ; ainsi que des protocoles dont l'objectif est la résilience. Pour ces derniers, nous considérons les trois solutions (RGBR, PRGBR et PRDGBR) proposées dans [4]. Ces algorithmes sont des variantes du protocole GBR, qui routent les messages suivant un arbre couvrant en largeur enraciné au puits. Ces trois variantes consistent à introduire de l'aléa et des duplications dans GBR. Notre étude montre que la résilience de SR3 est meilleure que celle de ces différents protocoles. De plus, contrairement à ces algorithmes, SR3 s'adapte face aux attaquants dont le comportement évolue au cours du temps.

2 Présentation de SR3

Nous considérons des réseaux connexes bidirectionnels, où tous les nœuds honnêtes ont régulièrement des données à router vers le puits. Chaque nœud dispose d'un identifiant unique, d'une clé partagée avec le puits, et peut se servir de cryptographie symétrique, de fonctions de hachage et de nonces. Les nœuds

connaissent également leurs voisins. Le code de notre algorithme, ainsi que les détails de notre modélisation et de nos preuves sont disponibles dans notre rapport technique [1].

Principe. Introduire de l'aléa dans un algorithme de routage est intéressant pour sa résilience car cela rend le routage imprévisible par l'attaquant. Ainsi, SR3 est conçu comme une marche aléatoire *renforcée*, se fondant sur un mécanisme de *réputation* pour calculer la probabilité de choisir un voisin au prochain saut. L'idée est d'augmenter la probabilité de router un message *via* un voisin si celui-ci se comporte bien. Pour cela, SR3 s'appuie sur des accusés de réception. Notre algorithme assure que pour chaque accusé de réception valide reçu, le message correspondant a bien été livré au puits. De cette manière, le nœud pourra légitimement augmenter sa confiance envers le voisin à qui il a envoyé le message en premier. Après un certain temps, tous les nœuds routeront de préférence leurs messages *via* leurs voisins de confiance. Ainsi, les messages auront tendance à suivre des routes qui les amènent sûrement au puits.

SR3 se fonde sur des nonces. Chaque message routé par SR3 est identifié par un nonce N_v, qui est généré par le nœud initiateur v. Le nœud v chiffre ce nonce avec les données à envoyer, à l'aide de la clé qu'il partage avec le puits, k_{vs}. Le chiffrement C et l'identifiant du nœud v sont ensuite routés vers le puits. Une fois le message validé, le puits produit l'accusé de réception $\langle N_v, v\rangle$. Ce dernier est renvoyé à l'émetteur initial v. Ainsi, à la réception de l'accusé, v est certain que le puits a reçu le message.

Cependant, un attaquant peut modifier à l'aveugle le contenu d'un paquet reçu. Pour éviter que le puits ne livre des informations erronées à l'application, nous ajoutons au message une *empreinte* H du nonce N_v, créée avec une fonction de hachage publique. À la réception d'un message, le puits déchiffre C avec la clé de l'émetteur prétendu pour obtenir N_v, et y applique la fonction de hachage. Le résultat doit être égal à H pour que le message soit considéré valide. De cette manière, si un attaquant altère le chiffrement C ou l'identité de l'émetteur d'un message, le puits détecte ces changements et rejette le message.

Mécanisme de réputation. À la réception d'un accusé de réception, le nœud v ayant initié le routage du message correspondant m peut conclure que m a été livré. Dans ce cas, v augmente la probabilité associée au voisin à qui m a été envoyé en premier. Pour ce faire, lors du premier envoi de m, v stocke le nonce de m et l'identifiant du premier destinataire dans une liste L_{Queue}. À la réception d'un accusé de réception, v vérifie s'il est le destinataire de l'accusé, puis si le nonce est contenu dans L_{Queue}. Dans ce cas, v récupère l'identifiant du voisin correspondant, augmente sa réputation et supprime l'entrée de L_{Queue}. Si v est le destinataire de l'accusé de réception, mais que la liste ne contient pas l'entrée correspondante, l'accusé est simplement ignoré.

À cause des limitations de mémoire, la taille de L_{Queue} est bornée à s_Q éléments. Quand un nœud doit router un message mais que la liste est pleine, l'élément le plus ancien est effacé. De cette manière les informations les plus récentes sont prioritaires. Nous remarquons également que les messages perdus ou dont l'accusé a été perdu finissent par être supprimés de la liste.

Enfin, il est possible qu'un message de données m retourne au nœud v qui l'a généré, à cause d'un cycle dans le réseau. Dans ce cas, la validité du message est vérifiée, puis le routage de m est réinitialisé et l'ancienne entrée dans L_{Queue} est remplacée par la nouvelle.

Évaluation de la réputation. Pour choisir le prochain nœud à qui va être envoyé un message, un nœud effectue un choix aléatoire parmi ses voisins, pondéré par leur réputation. La réputation d'un voisin correspond au nombre d'occurrences de son identifiant dans la liste $L_{Routing}$: à chaque réception d'un accusé de réception validant la livraison d'un message, le nœud initiateur augmente sa confiance envers le voisin choisi comme premier relais en ajoutant son identifiant dans cette liste.

Le choix du prochain saut suit la loi de probabilité suivante : pour un nœud v, soit X la variable aléatoire qui représente le voisin de v à qui envoyer le message. Soit δ_v le nombre de voisins de v, $|L_{Routing}|_x$ le nombre d'occurrences d'un nœud x dans $L_{Routing}$, et $|L_{Routing}|$ le nombre total d'éléments de cette liste. La probabilité de faire suivre un message à x est : $Pr(X = x) = \frac{|L_{Routing}|_x + \delta_v^{-1}}{|L_{Routing}|+1}$. Intuitivement, si un nœud doit router un message, il va choisir une valeur au hasard parmi $L_{Routing}$ ou un joker. Si un identifiant de nœud est tiré, le message lui sera transmis. Dans le cas du joker, un voisin sera choisi uniformément au hasard. Ainsi, plus le nœud fait confiance à un voisin, plus il lui transmettra de messages. Cependant, il y a toujours une probabilité positive de choisir un nœud sans considérer les réputations.

Afin d'assurer une résilience forte face aux attaquants qui changent de comportement au fil du temps, $L_{Routing}$ est une liste *FIFO* de taille bornée par s_R. Ainsi, les informations stockées seront toujours les plus

fraîches. De cette manière, si un attaquant se comporte bien dans un premier temps, sa réputation sera bonne. Si ensuite, il décide de perdre des messages, sa réputation baissera graduellement grâce aux accusés de réception des messages passant par d'autres chemins.

Routage des accusés de réception. Un accusé de réception n'est émis que si le message correspondant m a été livré par le puits. Nous pouvons donc supposer que le chemin suivi par m est sûr. Comme nos liens sont bidirectionnels, nous routons l'accusé de réception autant que possible suivant le chemin inverse de m.

Pour ce faire, chaque message de données laisse une trace de son passage dans tous les nœuds qui le relaient. Cette trace est stockée au niveau de chaque nœud dans la liste $L_{AckRouting}$: après chaque réception d'un message, les nœuds relais stockent l'empreinte du nonce du message et l'identifiant du voisin qui l'a relayé précédemment. Ces informations seront ensuite réutilisées pendant le routage des accusés de réception : quand un nœud v reçoit un accusé de réception, il calcule l'empreinte du nonce de cet accusé de réception, et cherche dans $L_{AckRouting}$ si le chemin correspondant est connu. Si c'est le cas, l'accusé de réception est relayé au nœud associé. Sinon, il est renvoyé à un voisin choisi uniformément au hasard.

Si un message de donnée boucle et revisite un nœud, l'information la plus pertinente sur sa provenance est la plus ancienne. Par conséquent, lors des ajouts dans $L_{AckRouting}$, le nœud doit toujours vérifier s'il possède déjà une information sur ce message : si c'est le cas, l'ancienne entrée est conservée.

Les accusés de réception peuvent cependant être perdus par des nœuds compromis. Afin d'éviter d'encombrer la mémoire des nœuds avec les informations de routage pour ces accusés, $L_{AckRouting}$ est de taille maximale s_A, et les ajouts dans une liste pleine suppriment les plus anciens éléments.

Enfin, un attaquant peut contrefaire un accusé de réception avec un faux destinataire. Ces faux accusés de réception peuvent être relayés indéfiniment. Puisqu'un nœud ne peut pas faire confiance à ses voisins, nous ne pouvons pas nous baser sur eux pour savoir si un accusé doit être relayé ou stoppé. Pour contourner ce problème, chaque nœud choisit de ne pas retransmettre un accusé de réception avec une probabilité $\frac{1}{N}$, où N est une borne supérieure sur le nombre de nœuds. Ainsi, un accusé de réception fera en moyenne N sauts avant d'être supprimé. Un avantage de ce mécanisme est qu'il favorise les routes courtes. En effet, les accusés de réception suivant des routes longues seront souvent supprimés avant d'atteindre leur destination. Or, la longueur de la route suivie par un accusé est corrélée à la longueur de la route suivie par le message correspondant. Ainsi, la réputation des voisins participant à des routes longues sera peu souvent renforcée.

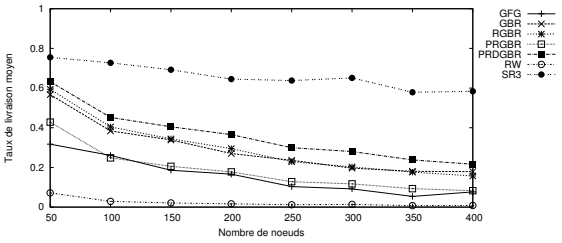
Sécurité. Nous avons analysé les propriétés de sécurité de SR3 à l'aide de *CryptoVerif* [2]. À partir de jeux créés en fonction du protocole, des propriétés désirées et des primitives cryptographiques utilisées, CryptoVerif détermine une borne sur la capacité d'un attaquant à casser ces propriétés. Cette borne dépend des tailles de paramètres et des propriétés des primitives cryptographiques utilisées. Nous avons prouvé trois propriétés de SR3 : premièrement, le protocole garantit la *confidentialité* des données routées. Deuxièmement, il est impossible de *contrefaire des messages de données* (cette propriété implique l'*authenticité* et l'*intégrité* de ces messages). Troisièmement, il est impossible de créer un accusé de réception pour un message qui n'a pas encore été livré au puits.

3 Analyse expérimentale

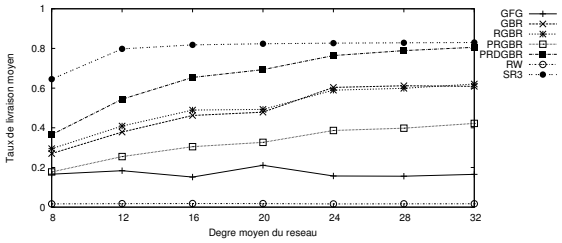
Méthodologie et paramètres. Nous avons évalué expérimentalement la résilience de SR3 face à RW, GFG, GBR et ses variantes (RGBR, PRGBR, PRDGBR) dans divers scénarios d'attaque. Nous avons utilisé *Sinalgo*, un simulateur de réseaux de capteurs sans fil. Nous ne présenterons ici que quelques résultats représentatifs, les autres étant disponibles dans le rapport technique [1].

Pour nos simulations, nous considérons des réseaux *à disque unitaire* connexes et générés en positionnant des nœuds uniformément au hasard sur une surface carrée. Les nœuds compromis sont choisis aléatoirement parmi les capteurs, et le puits est placé au centre de l'aire de simulation. Les communications sont asynchrones et *FIFO*, les temps de transfert suivent une distribution exponentielle, de même que l'intervalle entre deux générations de message pour un nœud. Nos simulations durent le temps de traiter 500 000 messages, et nous testons 20 topologies pour chaque expérience. SR3 requiert quatre paramètres : N (une borne supérieure sur le nombre de nœuds du réseau), et les bornes sur les tailles de chaque liste (s_R, s_Q et s_A). Dans chaque simulation, N est fixé au nombre de nœuds dans le réseau. Nous fixons respectivement s_R, s_Q et s_A à 10, 3 et 5 éléments. Ces valeurs ont été déterminées par une évaluation expérimentale détaillée dans le rapport technique [1], et garantissent de bonnes performances en gardant la consommation mémoire faible.

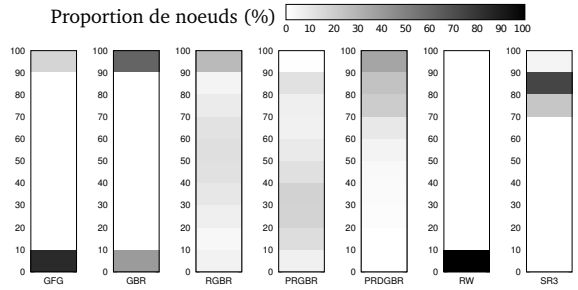
Résultats. La figure 1a présente les taux de livraison de messages observés dans les réseaux de degré moyen



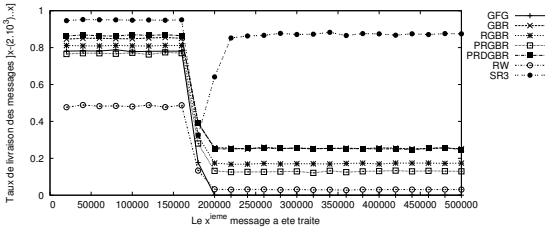
(a) Taux de livraison moyen, 30% BH, $\overline{\delta} = 8$



(b) Taux de livraison moyen, 30% BH, $n = 200$



(c) Distribution des taux de livraison, 30% BH, n=200, $\overline{\delta}$=32



(d) Taux de livraison moyen par paquets de 20 000 messages, 5% WH→BH, 5% BH, $n = 200$, $\overline{\delta} = 8$

$\overline{\delta} = 8$, confrontés à 30% de *blackholes* (BH), *i.e.*, des nœuds compromis qui perdent tous les messages qu'ils reçoivent. En faisant varier le nombre de nœuds dans le réseau, on remarque que SR3 donne toujours un taux de livraison supérieur aux autres protocoles de notre panel. L'écart se creuse encore dans les grands réseaux. La figure 1b montre les taux de livraison de messages observés dans les réseaux de 200 nœuds, confrontés à 30% de *blackholes*, en faisant varier $\overline{\delta}$ de 8 à 32. Encore une fois, SR3 offre les meilleures performances. De plus, comme RW et GFG, nous remarquons que SR3 est peu sensible aux variations de $\overline{\delta}$. Au contraire, les taux observés pour GBR et ses variantes sont faibles dans les réseaux de faible densité. Dans les réseaux très denses, les performances de PRDGBR se rapprochent de SR3. Cependant, PRDGBR duplique les messages à chaque saut, ce qui entraîne un surcoût de communication important.

Nous avons ensuite mesuré l'*équité* de SR3, *i.e.*, à quel point les nœuds honnêtes ont des taux de livraison comparables. La figure 1c présente la distribution des taux de livraison des nœuds dans des réseaux de 200 nœuds et de degré moyen 32. GBR n'est pas équitable, puisqu'on distingue clairement deux classes de nœuds : ceux dont tous les messages sont livrés et ceux dont tous les messages sont perdus. À l'inverse, SR3 est équitable, puisque presque tous les nœuds ont un taux de livraison proche de 80%.

Nous avons ensuite évalué l'*adaptativité* de SR3. La figure 1d présente le taux de livraison moyen par fenêtre glissante de 20 000 messages dans un réseau de 200 nœuds et de degré moyen 8, face à 5% de *blackholes* et 5% de *wormholes*, qui se comportent initialement comme des nœuds directement connectés au puits (générant une excellente réputation), puis comme des *blackholes* après le premier tiers de la simulation. Notre protocole récupère effectivement un bon taux de livraison rapidement après le changement de comportement, ce qui n'est pas le cas pour les autres protocoles.

Enfin, nous avons constaté que des attaquants ne bloquant qu'une partie des messages (*selective forwarding*), ou déclarant plusieurs identités (nœuds *Sybil*) n'ont pas spécialement d'impact sur notre protocole par rapport aux autres présentés ici. Nous avons également évalué les performances de SR3 dans les réseaux sans attaquants, et nos résultats indiquent des routes de longueur raisonnable : par exemple, dans des réseaux de 400 nœuds et de degré moyen 8, les routes ont une longueur moyenne de 20 sauts.

Références

[1] K. Altisen, S. Devismes, R. Jamet, and P. Lafourcade. SR3 : A secure and resilient reputation-based routing protocol. Technical report, Verimag, January 2013. `http://www-verimag.imag.fr/~rjamet/SR3/`.

[2] B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Trans. Dependable Sec. Comput.*, 5(4) :193–207, 2008.

[3] T. Eisenbarth and S. Kumar. A survey of lightweight-cryptography implementations. *Design & Test of Computers, IEEE*, 24(6) :522–533, 2007.

[4] O. Erdene-Ochir, A. Kountouris, M. Minier, and F. Valois. Enhancing resiliency against routing layer attacks in wireless sensor networks : Gradient-based routing in focus. *International Journal On Advances in Networks and Services*, 4(1, 2) :38–54, 2011.