Spesso, per l’erogazione di un servizio in rete, è necessaria da parte di un utente la divulgazione dei propri dati personali. A volte è sufficiente il coinvolgimento di una sola entità, il service provider (SP), mentre in altri casi i dati possono essere trasmessi a terze parti per l’ottenimento del servizio, o per finalità di mercato. Tali informazioni riservate possono viaggiare anche molto lontano, finendo in Paesi diversi da quello di partenza, processate da aziende con sistemi così complessi da perdere infine la cognizione di quali dati siano effettivamente stati raccolti e condivisi fra SP.

In questo scenario si aprono diversi spunti riguardo all’utilizzo dei dati personali: la trasmissione (anche solo parziale) degli stessi era autorizzata? Le politiche per il rispetto della privacy alle quali il DO ha aderito sono le stesse all’interno delle terze parti? Quali fattori aggiuntivi entrano in gioco (leggi nazionali ed internazionali, politiche aziendali…)?

Le sticky policies si propongono come strumento in grado di rispondere a queste necessità: si tratta di metadati che specificano l’utilizzo consentito dei dati a cui fanno riferimento, ed ai quali “restano attaccate” (da cui il nome). Sono espresse in un formato machine readable, che ne permette l’automatica lettura e applicazione nella granularità desiderata.

L’architettura necessaria per realizzare questa opportunità si compone, nella sua versione più scarna, di tre componenti: il proprietario dei dati (DO), una terza parte interessata ai dati (spesso un SP) ed un’autorità con un grado di affidabilità variabile. Questa struttura non è nuova in quanto è già stata realizzata ed è in uso per quanto riguarda l’infrastruttura a chiave pubblica; inoltre, una implementazione simile è stata prevista all’interno dello standard XACML (standard2005extensible). Il linguaggio XACML definisce uno standard per l’implementazione dell’Attribute-Based Access Control grazie ad una infrastruttura che distribuisce i compiti di creazione, memorizzazione, verifica ed applicazione delle politiche scelte. Fra di esse, l’entità di enforcement è quella che più si adatta ed è necessaria nel contesto delle sticky policies, supponendo che sia il DO a formulare le politiche di divulgazione e utilizzo dei suoi dati.

Ciò richiederebbe però che tale PEP si occupasse anche dei compiti strettamente relativi agli aspetti crittografici o di sicurezza come gestione e pubblicazione delle chiavi in caso di PKE, generazione delle chiavi private ed identificazione in caso di IBE, eccetera. In altre parole, dovrebbe essere completamente fidata, costituire uno storage sicuro, essere sempre online e disponibile, introducendo all’interno dell’architettura un single point of failure. Per sgravare il PEP da alcuni di questi compiti, diminuendo quindi il livello di affidabilità richiesto, è possibile prevedere una ulteriore entità chiamata Trusted Third Party (tang2008using): TTP mantiene il ruolo di gestore delle chiavi e di attuatore delle verifiche dell’identità, mentre il PEP si limita al controllo delle policy per la divulgazione dei dati. Ciò aumenta la sicurezza in quanto l’affidabilità richiesta per il PEP è minore.

Nello stesso articolo, Tang fornisce un’analisi completa delle attuali tecniche disponibili per l’implementazione di sticky policies: PKI, IBE, (CP-) ABE e PRE, suggerendo una quinta possibilità, TPRE, per migliorare l’efficienza di PRE. Vediamone alcune in dettaglio.

L’articolo pearson2011sticky ed il progetto **EnCoRe** mostrano una possibile implementazione di SP tramite l’utilizzo di politiche espresse in formato XML e architettura a chiave pubblica con cifrario ibrido per l’invio delle politiche insieme ai dati offuscati. L’aderenza delle policies ai dati viene ottenuta mediante la cifratura della chiave simmetrica insieme all’hash delle stesse: il messaggio sarà poi decifrato da una terza parte fidata (TA) che si occupa anche di verificare l’affidabilità del SP che richiede le informazioni. Come è possibile intuire, questa soluzione richiede l’esistenza di una infrastruttura per il supporto di certificati e l’erogazione di chiavi per cifratura e firma; essa riveste anche il ruolo di garante delle policies (deve verificare l’attendibilità del SP prima di rilasciare la chiave simmetrica di sessione K) e mantenere un sistema di logging sicuro a questo scopo. In questo contesto, ogni utente che volesse tutelare la propria privacy deve prima ottenere una coppia di chiavi.

È possibile aggiungere alle entità in gioco uno storage sicuro che si occupi di mantenere il dato offuscato al posto del singolo DO.

Il protocollo di scambio dei dati proposto in questo articolo lega la policy ai dati per mezzo della cifratura con chiave pubblica della chiave simmetrica; la stessa operazione garantisce l’integrità della policy per mezzo della funzione di hashing. Infine, questo dato cifrato viene firmato dall’utente proprietario, per garantirne la paternità. Dal protocollo si osserva che non è possibile modificare il formato dei messaggi cambiando la chiave di sessione o le policies, mentre sarebbe possibile effettuare attacchi di replay, anche se di poca utilità, per l’assenza di timestamps nel protocollo. Sarebbe possibile spacciarsi per la TA presso il SP, intercettando il messaggio legittimo e inserendo nel messaggio di risposta una chiave contraffatta (per impedire la cifratura dei dati, o produrre collisioni in caso siano note, o ancora inserire nel messaggio iniziale dati contraffatti (**not really sure about this**), decifrabili con la chiave fasulla.

In generale, questo scenario prevede l’interazione con una singola TA per il rilascio dei dati, mentre il coinvolgimento di un numero maggiore di entità fidate può migliorare il livello di sicurezza garantito, impedendo collusioni o violazioni. Si vedrà più avanti che ciò è possibile mediante l’utilizzo di uno schema di threshold cryptography (shamir1979share), menzionato come alternativa insieme alla identity-based cryptography (shamir1984identity).

Rimane inoltre come punto aperto l’affidabilità dei SP: una volta infatti che la TA ha rilasciato la chiave di decifrazione dei dati sensibili, non vi è più alcun modo di controllare l’utilizzo che ne viene fatto, e l’applicazione delle politiche di protezione della privacy è lasciata alle “buone intenzioni” dello specifico SP. Altri punti aperti riguardano il costo computazionale relativo alla gestione dell’infrastruttura PKI con le aggiunte necessarie e l’emissione di certificati per ogni utente che volesse avvalersi di sticky policies. Infine, si deve anche considerare la necessità di provare l’affidabilità di una macchina remota (si intende che la macchina si comporta sempre come desiderato, per lo scopo voluto). A tale scopo sono stati sviluppati alcuni moduli hardware che hanno una funzione di “root of trust” e che permettono di garantire l’affidabilità e l’integrità della macchina e del software in funzione. Un dispositivo di questo tipo è basato su Trusted Platform Module (standard2009trusted).

La possibilità dell’utilizzo di IBE viene esplorata in mont2003towards, che a sua volta si propone di risolvere alcune “open issues” come la stickyness dei dati e l’utilizzo di un maggior numero di TA.

L’IBE (shamir1984identity) è una tipologia di schema crittografico grazie al quale è possibile sia firmare che cifrare dati. Le sue potenzialità rispetto all’infrastruttura PKI sono l’eliminazione dell’infrastruttura stessa (chiavi pubbliche e private, certificati, …) a favore di un sistema in cui la chiave pubblica è un attributo univoco non ripudiabile (come ad esempio l’indirizzo email) e la chiave privata viene generata su richiesta, in seguito ad identificazione, da un’entità fidata detta PKG (Private Key Generator).

Ad oggi esistono alcune implementazioni di IBE: alcune di utilizzo commerciale (sviluppate da HPE Security (IBEimplVoltage) basate sullo standard (boyen2007identity) ), altre per scopi universitari e di ricerca (Lynn2013, ducas2016 ).

In questo scenario, si sceglie di usare come chiave di cifratura una stringa che costituisce la sticky policy stessa: ciò realizza la stickyness tramite associazione univoca tra dati cifrati e chiave, permettendo anche di preservare l’integrità dei dati. La TA sarà poi in grado di generare la corrispondente chiave di decifrazione.

L’utilizzo di IBE permette il coinvolgimento di più di una TA al fine di ottenere la divulgazione dei dati personali, in quanto i dati non sono più legati obbligatoriamente ad una sola TA a causa della chiave. In particolare ciò può essere ottenuto mediante l’utilizzo di un threshold scheme (shamir1979scheme, sahai2005fuzzy): la chiave per decifrare il dato, o eventualmente il dato stesso, viene divisa in un numero n di parti, ed almeno k TA devono completare il processo di verifica per rilasciare la chiave e permettere la decifrazione del dato. L’utilizzo di questo schema comporterebbe l’innalzamento dei livelli di sicurezza, pur richiedendo un impegno computazionale maggiore per la creazione dello schema.

Il protocollo usato è, ad alto livello, lo stesso presentato nel precedente paper: l’EU procede alla scelta di una policy e all’offuscamento dei dati, che trasmette poi al SP richiedente. Questi inizia una procedura di identificazione presso la TA al fine di ricevere la chiave di decifrazione. Anche in questa soluzione si fa esplicitamente uso di moduli trusted computing che permettano di verificare l’affidabilità della piattaforma hardware e del software remoto e di un sistema di logging affidabile. Non viene risolta la problematica della pericolosità dell’utilizzo dei dati dopo la loro decifrazione.

La soluzione proposta è però di troppo alto livello, in quanto presenta alcune possibili vulnerabilità: la cifratura per mezzo della policy è facilmente replicabile o riutilizzabile (Tang nel suo tang2008using suggerisce un esempio di ID||timestamp), e in caso di rottura del cifrario rende la policy inutilizzabile, in quanto si richiede il cambio della chiave. Tuttavia questo approccio è quello che permette, insieme a CP-ABE, la maggior aderenza delle policies ai dati.

Inoltre, l’articolo di Mont, Pearson e Siani non prende in esame le tecniche di Proxy Reencryption (green2007identity) e di cyphertext-policy Attribute-Based Encryption (bethencourt2007ciphertext) esaminate invece da Tang. Il secondo caso presenta evidenti vantaggi in quanto parte delle condizioni da soddisfare è già verificata al momento della generazione della chiave di ogni destinatario, che potrà decifrare dei messaggi solo se possiede determinati attributi. In questo senso, questa tecnica permette di realizzare un controllo a grana molto più fine, ma presenta un notevole svantaggio in caso sia necessario sostituire una chiave privata, poiché questa non si riferisce ad un singolo utente ma ad una categoria di utenti (e di relativi messaggi).

La PRE è invece adatta al caso delle sticky policies in quanto permette maggiore interoperabilità fra tecniche crittografiche (PKE e IBE) ed elimina la necessità di uno storage sicuro. Come nel caso di PKE, la stickyness è realizzata attraverso il PEP, che controlla la validità delle condizioni imposte prima di procedere alla reencryption per il destinatario. Questo schema è però più efficiente dello schema PKE nel caso in cui lo stesso testo vada mandato a destinatari diversi, poiché il mittente si occupa di una sola cifratura, e sarà poi il PEP ad occuparsi, se e quando necessario, di cifrarlo per lo specifico destinatario.

Inoltre, applicazioni simili a quella delle sticky policies sono mostrate in (green2007identity) come possibili future applicazioni della proxy reencryption. In particolare si sottolineano due proposte: la prima, discussa anche in altri paper (), riguarda lo storage di files cifrati, che elimina la necessità di uno storage sicuro. La seconda assomiglia a quanto detto precedentemente per Mont (mont2003towards): si tratta di decifrare dei dati inserendo, oltre all’identità del destinatario, anche altri attributi necessari alla decifrazione, come l’appartenenza ad un intervallo temporale o il possesso di determinate autorizzazioni di accesso.